Napoleão Povoa Ribeiro Filho

Melhorando o desempenho da técnica de clusterização hierárquica Single Linkage utilizando a Metaheurística Grasp

Ano 2023

Napoleão Povoa Ribeiro Filho

Melhorando o desempenho da técnica de clusterização hierárquica Single Linkage utilizando a Metaheurística Grasp

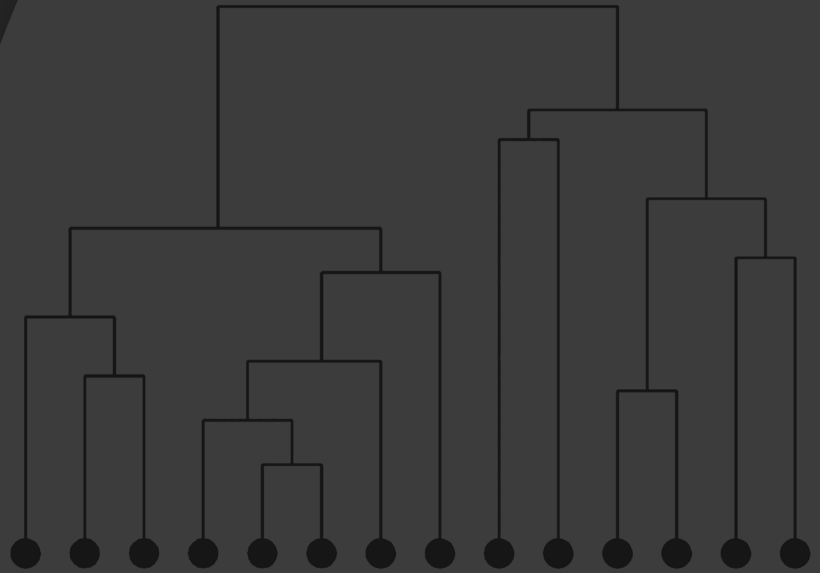
Ano 2023

CIÊNCIAS EXATAS E DA TERRA E ENGENHARIAS

Melhorando o desempenho da técnica de clusterização hierárquica Single Linkage utilizando a Metaheurística GRASP

Diagramação: Letícia Alves Vitral
Correção: Maiara Ferreira
Indexação: Amanda Kelly da Costa Veiga
Revisão: O autor
Autor: Napoleão Póvoa Ribeiro Filho

Dados Internacionais de Catalogação na Publicação (CIP)

R484 Ribeiro Filho, Napoleão Póvoa
Melhorando o desempenho da técnica de clusterização hierárquica Single Linkage utilizando a Metaheurística GRASP / Napoleão Póvoa Ribeiro Filho. – Ponta Grossa - PR: Atena, 2023.

Formato: PDF
Requisitos de sistema: Adobe Acrobat Reader
Modo de acesso: World Wide Web
Inclui bibliografia
ISBN 978-65-258-1024-9
DOI: https://doi.org/10.22533/at.ed.249231702

1. Algorítmos. I. Ribeiro Filho, Napoleão Póvoa. II. Título.

CDD 518.1

Elaborado por Bibliotecária Janaina Ramos – CRB-8/9166

Atena Editora
Ponta Grossa – Paraná – Brasil
Telefone: +55 (42) 3323-5493
www.atenaeditora.com.br
contato@atenaeditora.com.br

DECLARAÇÃO DO AUTOR

O autor desta obra: 1. Atesta não possuir qualquer interesse comercial que constitua um conflito de interesses em relação ao conteúdo publicado; 2. Declara que participou ativamente da construção dos respectivos manuscritos, preferencialmente na: a) Concepção do estudo, e/ou aquisição de dados, e/ou análise e interpretação de dados; b) Elaboração do artigo ou revisão com vistas a tornar o material intelectualmente relevante; c) Aprovação final do manuscrito para submissão.; 3. Certifica que o texto publicado está completamente isento de dados e/ou resultados fraudulentos; 4. Confirma a citação e a referência correta de todos os dados e de interpretações de dados de outras pesquisas; 5. Reconhece ter informado todas as fontes de financiamento recebidas para a consecução da pesquisa; 6. Autoriza a edição da obra, que incluem os registros de ficha catalográfica, ISBN, DOI e demais indexadores, projeto visual e criação de capa, diagramação de miolo, assim como lançamento e divulgação da mesma conforme critérios da Atena Editora.

DECLARAÇÃO DA EDITORA

RESUMO

O problema de clusterização (agrupamento) consiste em, a partir de uma base de dados, agrupar os elementos de modo que os mais similares fiquem no mesmo *cluster* (grupo), e os elementos menos similares fiquem em *clusters* distintos. Há várias maneiras de se realizar esses agrupamentos. Uma das mais populares é a hierárquica, onde é criada uma hierarquia de relacionamentos entre os elementos. Há vários métodos de se analisar a similaridade entre elementos no problema de clusterização. O mais utilizado entre eles é o método single linkage, que agrupa os elementos que apresentarem menor distância entre si. Para se aplicar a técnica em questão, uma matriz de distâncias é a entrada utilizada. Esse processo de agrupamento gera ao final uma árvore invertida conhecida como dendrograma. O coeficiente de correlação cofenética (*ccc*), obtido após a construção do dendrograma, é utilizado para avaliar a consistência dos agrupamentos gerados e indica o quão fiel o dendrograma está em relação aos dados originais. Dessa forma, um dendrograma apresenta agrupamentos mais consistentes quando o *ccc* for o mais próximo de um (1). O problema de clusterização em todas as suas vertentes, inclusive a clusterização hierárquica (objeto de estudo nesse trabalho), pertence a classe de problemas NP-Completo. Assim sendo, é comum o uso de heurísticas para obter soluções de modo eficiente para esse problema. Com o objetivo de gerar dendrogramas que resultem em melhores *ccc*, é proposto no presente trabalho um novo algoritmo que utiliza os conceitos da metaheurística GRASP. Também é objetivo deste trabalho implementar tal solução em computação paralela em um cluster computacional, permitindo assim trabalhar com matrizes de dimensões maiores. Testes foram realizados para comprovar o desempenho do algoritmo proposto, comparando os resultados obtidos com os gerados pelo software R.

Palavras-chave: GRASP, Clusterização Hierárquica, Coeficiente de Correlação Cofenética.

ABSTRACT

The problem of clustering (grouping) consists of, from a database, group the elements so that more queries are in the same cluster (group) and less similar elements are different clusters. There are several ways to accomplish these groupings. One of the most popular is the hierarchical, where a hierarchical relationships between the elements is created. There are several methods of analyzing the similarity between elements in the clustering problem. The most common among them is the single linkage method, which brings together the elements that are experiencing less apart. To apply the technique in question, distance matrix is the input used. This grouping process generates the end an inverted tree known as dendrogram. The cophenetic correlation coefficient (*ccc*), obtained after the construction of the dendrogram is a measure used to evaluate the consistency of the clusters generated and indicates how faithful he is in relation to the original data. Thus, a dendrogram gives more consistent clusters when the *ccc* is closer to one (1). The clustering problem in all its aspects, including hierarchical clustering (object of study in this work), belongs to the class of NP-complete problems. Therefore, it is common to use heuristics for efficient solutions to this problem. In order to generate dendrograms that result in better *ccc*, it is proposed in this paper a new algorithm that uses the concepts of GRASP metaheuristic. It is also objective of this work to implement such a solution in parallel computing in a computer cluster, thus working with arrays larger. Tests were conducted to confirm the performance of the proposed algorithm, comparing the results with those generated by the software R.

Keywords: GRASP, Hierarchical clustering, Cophenetic Correlation Coefficient.

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO 1

Visão Geral do Problema 3

Justificativa 3

Objetivo Geral 4

Objetivos específicos 4

Organização da Dissertação 4

CAPÍTULO 2

PROBLEMA DE CLUSTERIZAÇÃO HIERÁRQUICA 6

Matriz de distâncias 7

Coeficiente de Correlação Cofenética 8

Métodos Aglomerativos 9

Aplicações Práticas da Clusterização Hierárquica 11

Clusterização para formação de Células de Manufatura (PFCM) 11

Clusterização aplicada ao Mercado Financeiro Brasileiro 13

Clusterização aplicada à bioinformática 14

CAPÍTULO 3

HEURÍSTICA GRASP PROPOSTA 17

Fase de Construção 18

Fase de Busca Local 19

CAPÍTULO 4

PROGRAMAÇÃO PARALELA 20

O Padrão MPI 22

Conceitos básicos de processamento paralelo 22

Operações Coletivas no processamento paralelo 23

Etapas para execução em um ambiente de processamento pa- ralelo 24

MPJ Express 25

GRASP Paralelo 26

CAPÍTULO 5

RESULTADOS E DISCUSSÕES 28

As Instâncias e o Ambiente de Teste da solução GRASP 29

Resultados obtidos pela solução GRASP Sequencial 30

As Instâncias e o Ambiente de Teste da solução GRASP Paralela 31

Resultados obtidos pela solução GRASP Paralela.. 33

CAPÍTULO 6

CONCLUSÕES E TRABALHOS FUTUROS 39

REFERÊNCIAS 41

APÊNDICES... 45

APÊNDICE A – Pôster publicado: XLVII SBPO 45

SOBRE O AUTOR 46

SUMÁRIO

INTRODUÇÃO

A necessidade de classificar elementos em grupos por suas características está presente em várias áreas do conhecimento, como nas ciências biológicas, ciências sociais e comportamentais, ciências da terra, medicina, informática, entre outras (JAIN; MURTY; FLYNN, 1999). O trabalho de (HARTIGAN, 1975) oferece uma visão bem ampla de vários estudos publicados acerca da utilização de técnicas de análise de agrupamentos. Na área médica, por exemplo, agrupamento de doenças por sintoma ou curas pode levar a taxonomias muito úteis, como apresentado em (MAKRETSOV et al., 2004). Em áreas da psiquiatria, por exemplo, considera-se que o agrupamento de sintomas como paranóia, esquizofrenia e outros é essencial para se determinar a terapia adequada (LASTRA et al., 2000). Na arqueologia, por outro lado, também se tem tentado agrupar civilizações ou épocas de civilizações com base em ferramentas de pedra, objetos funerários, etc (ALDENDERFER, 1982). No processo de mineração de dados, a funcionalidade de segmentar um conjunto de dados em subgrupos homogêneos também é muito utilizada (HAN; KAMBER; PEI, 2011). De forma geral, toda vez que se faz necessário classificar uma grande quantidade de dados desconhecidos em grupos menores e mais compreensíveis, são utilizados métodos de agrupamento (CASTRO; PRADO, 2001).

A clusterização busca agrupar elementos de dados, formando grupos, baseando-se na similaridade ou dissimilarirade contidas em uma amostra de dados (ANDERBERG, 2014). Os grupos são determinados de forma a obter homogeneidade interna e heterogeneidade externa.

Já a clusterização hierárquica em específico, como descrito em (JOHNSON, 1967), consiste em uma série de agrupamentos sucessivos de elementos, gerando no final desse processo um diagrama bi-dimensional chamado de dendrograma ou diagrama de árvore, que mostra como os agrupamentos estão relacionados. Também é possível, nessa técnica, identificar os elementos que formam cada cluster (grupo) e qual o valor de similaridade que gerou cada agrupamento, por exemplo. Pode-se entender, dessa forma, o agrupamento hierárquico como sendo uma especificação de ações que resultam em uma estrutura de dados.

A correlação cofenética de um dendrograma é a correlação linear entre a matriz de similaridade utilizada para construir o dendrograma e a matriz de similaridade cofenética, obtida a partir do dendrograma (WEBB, 2003).Nesse contexto, o coeficiente de correlação cofenética (*ccc*), introduzido por (SOKAL; ROHLF, 1962), tem sido amplamente utilizado em vários estudos para medir a qualidade do dendrograma construído pelo algoritmo hierárquico. Para (FARRIS, 1969), tal coeficiente indica o grau de ajuste de uma

classificação de um grupo de dados. Isso quer dizer que o ccc serve para medir a qualidade de um dendrograma e assim, indicar o grau de distorção com relação a matriz de dados original. Um alto valor desse coeficiente, ou seja, próximo de um, indica que o dendrograma apresenta baixo grau de distorção e representa corretamente as estruturas latentes no conjunto de dados (METZ, 2006).

De acordo com (HRUSCHKA, 2001), o problema de clusterização, que envolve a identifcação de um conjunto de categorias ou grupos, é classificado como NP-Completo, limitando com isso o uso de métodos exatos. A tentativa de se encontrar uma solução ótima globalmente é usualmente inviável sob o ponto de vista computacional (ARABIE; HUBERT, 1996). Por isso, a grande maioria dos métodos existentes na literatura são métodos aproximados ou heurísticas.

Devido à grande heterogeneidade das aplicações de problemas de clusterização, as heurísticas utilizadas são normalmente desenvolvidas para determinadas classes de problemas, ou seja, não existe uma heurística genérica que seja a melhor em todas as aplicações de clusterização (DIAS; OCHI, 2004).

Agora, com relação as metaheurísticas ou heurísticas denominadas inteligentes, de acordo com (TRINDADE; OCHI, 2004), somente a partir dos anos 80 elas começaram a surgir na literatura. Tais como Algoritmos Genéticos, propostos por (HOLLAND, 1992), *Greedy Randomized Adaptive Search Procedure* (GRASP) proposto por (FEO; RESENDE, 1995), entre outros. As metaheurísticas nada mais são do que um processo iterativo ou de refinamento de solução aplicado a um determinado problema. Tal processo guia uma heurística subordinada, pela combinação de diferentes conceitos, podendo manipular uma solução completa, incompleta ou um conjunto de soluções. O seu objetivo é, de acordo com (ALVARENGA; ROCHA, 2006), através de mecanismos que permitam escapar de um ótimos locais pobres, obter um desempenho melhor que as heurísticas tradicionais. Isso quer dizer que as metaheurísticas buscam melhorar o resultado de heurísticas subordinadas alterando seu comportamento guloso, onde a cada iteração a mesma decisão é tomada, esperando assim obter um resultado que seja próximo do melhor resultado possível de ser alcançado. Contudo, para se saber se o resultado alcançado foi o melhor possível, é preciso testar todas as possibilidades. Entretanto, isso não é possível de ser feito dentro de um tempo polinomial, o que caracteriza tais problemas com complexidade computacional do tipo np- completo. Nesses casos são utilizadas as metaherísticas para tentar melhorar os resultados das heurísticas utilizadas como base, através da modificação de seu comportamento guloso.

Com intuito de gerar um dendrograma que possa alcançar um coeficiente de correlação cofenética, em um tempo computacional razoável, melhor que o gerado pelo método

tradicional *single linkage*, é proposta neste trabalho a aplicação de uma metaheurística GRASP para o problema de clusterização hierárquica.

VISÃO GERAL DO PROBLEMA

A clusterização hierárquica é utilizada para gerar agrupamentos (subgrupos) que sejam internamente homogêneos e externamente heterogêneos entre si. De acordo com (JAIN; MURTY; FLYNN, 1999), o método mais popular e comumente mais utilizado para gerar tais agrupamentos é o *single linkage*. Tal método considera a menor distância entre os elementos como critério. A saída do processo de agrupamento hierárquico é representada na forma de árvore invertida, conhecida como dendrograma. Para avaliar a consistência de tais agrupamentos, comumente é utilizado o coeficiente de correlação cofenética. Tal correlação mede simplesmente o grau de associação entre os elementos da matriz original de dados (fenética) e a matriz gerada após a aplicação do método de agrupamento (cofenética). Assim, quanto mais próximo de um (1) for o valor do coeficiente, melhor é a representação gerada para um agrupamento.

JUSTIFICATIVA

Uma abordagem gulosa de um algoritmo é caracterizada por tomar uma decisão baseada sempre no mesmo critério, a cada iteração. E ao final de sua execução, espera-se que o resultado obtido seja o melhor possível. O método de funcionamento de tais algoritmos e seus resultados o caracterizam como heurísticos. Porém, só se saberá se o resultado obtido pelo algoritmo guloso foi o melhor possível se forem verificadas todas as possibilidades. Esse resultado obtido pelo algoritmo guloso é chamado de ótimo local. E o melhor resultado possível, considerando todos os resultados existentes para o problema, é chamado de ótimo global.

O algoritmo de clusterização hierárquica *single linkage* é um algoritmo guloso que considera sempre a menor distância entre dois pontos para agrupá-los. Ao final da execução de tal algoritmo, é gerado um dendrograma, que é uma representação gráfica dos clusters formados. Com o objetivo de identificar a qualidade do dendrograma gerado e seu nível de distorção com relação aos dados utilizados como entrada para o algoritmo *single linkage*, comumente é utilizado o coeficiente de correlação cofenética.

De acordo com (RESENDE; RIBEIRO, 2014), a metaheurística GRASP é definida como “...uma metaheurística do tipo *multi-start* para problemas de otimização combinatória, no qual cada iteração consistem basicamente de duas fases: a construção de uma solução inicial de forma gulosa, aleatória e adaptativa e um procedimento de busca local que tem

o objetivo de aprimorar a solução gerada na primeira fase. Cada iteração GRASP trabalha de forma independente e o resultado final será então a melhor solução dentre todas as iterações '. Com o intuito de melhorar os dendrogramas obtidos pelo método de clusterização hierárquica *single linkage*, é proposto nesse trabalho a utilização da metaheurística GRASP. A utilização dessa metaheurística possibilita a exploração de outras soluções do universo de soluções possíveis. Isso é realizado, instruindo a heurística subordinada (nesse caso, *single linkage*), a tomar decisões diferentes a cada iteração, adaptando-a para funcionar como a fase construtiva do método, e após refinando-a no processo de busca local. Ao final de cada iteração do algoritmo, é calculado o *ccc* para verificar a qualidade do dendrograma obtido e a melhor solução de todas as iterações é armazenada como solução final.

OBJETIVO GERAL

Implementar um algoritmo, utilizando a metaheurística GRASP, que seja capaz de gerar agrupamentos hierárquicos com coeficientes de correlação cofenética melhores que os gerados pelo método *single linkage*.

Objetivos específicos

- Implementar uma metaheurística GRASP para o problema de clusterização hierárquica, fazendo uso de uma versão alterada da heurísitica *single linkage* como método construtivo.
- Realizar testes que comprovem que o algoritmo proposto consegue alcançar coeficientes de correlação cofenética melhores que os obtidos pelo método *single linkage*.
- Implementar uma versão do algoritmo proposto para trabalhar em um ambiente computacional paralelo e distribuído, de modo a poder melhorar o desempenho do processo de clusterização quando um grande volume de dados são utilizados como entrada.

ORGANIZAÇÃO DA DISSERTAÇÃO

Essa dissertação encontra-se organizada em 6 capítulos, incluindo esta introdução. A seguir, apresenta-se uma breve descrição de cada um dos capítulos subsequentes:

- Capítulo 2: apresenta uma introdução ao problema de clusterização hierárquica bem como exemplos de suas aplicações práticas.
- Capítulo 3: discorre a respeito da metaheurística GRASP desenvolvida para o problema em questão e seus componentes.
- Capítulo 4: aborda a questão da programação paralela em um agrupamento de computadores e trata da solução adotada nesse trabalho.

- Capítulo 5: são mostrados resultados dos testes computacionais obtidos a partir da técnica proposta, e também é feita uma discussão sobre os resultados alcançados.
- Capítulo 6: são apresentadas as conclusões e trabalhos futuros.

PROBLEMA DE CLUSTERIZAÇÃO HIERÁRQUICA

Existem diversas abordagens de agrupamentos (*clustering*), como probabilística, otimização e hierárquica (JAIN; MURTY; FLYNN, 1999). Cada abordagem utiliza uma maneira própria para identificar e representar os agrupamentos. Nesse trabalho, os agrupa- mentos serão obtidos através da clusterização hierárquica, onde os *clusters* são representados por uma estrutura conhecida como dendrograma, que é um tipo especial de árvore onde os grupos vão sendo formados de baixo para cima até restar um único grupo. De acordo com (FESTA; RESENDE, 2009), cada método aglomerativo escolhido influenciará nos detalhes do dendrograma gerado. Em tal estrutura, como pode ser verificado na Figura 1, na horizontal aparecem os elementos, de acordo com os grupos formados. Na vertical, é mostrada a altura onde ocorreu a formação do agrupamento.

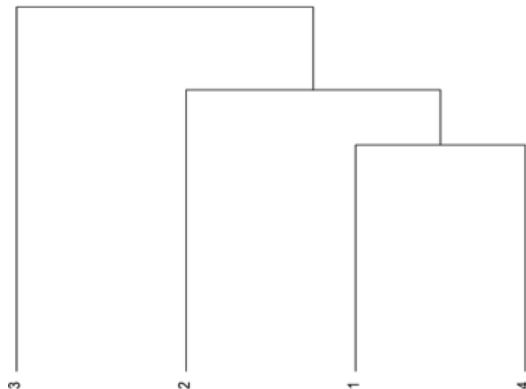



Figura 1: Características de 4 elementos a serem considerados no agrupamento.

Nesse caso, se está falando da abordagem conhecida como *bottom-up* ou aglomerativa, onde cada dado inicial representa um *cluster*. Tais *clusters* vão sendo recursivamente agrupados, considerando alguma medida de similaridade adotada, até que todos os dados façam parte de um único agrupamento. Tendo em vista os dados da Tabela 1, com 4 elementos com 4 características cada, tem-se um dendrograma como o da Figura 1.

CARACTERÍSTICAS	**INDIVÍDUOS**			
	Amostra 1	**Amostra 2**	**Amostra 3**	**Amostra 4**
X	22	24	20	26
Y	20	19	22	25
Z	24	20	28	23
W	21	26	24	25

Tabela 1: Características de 4 elementos a serem considerados no agrupamento.

Para melhor compreensão de todas as etapas executadas por um algoritmo de

clusterização hierárquica, a seção a seguir discorrerá sobre a matriz de distâncias, que é a entrada utilizada por tais algoritmos.

MATRIZ DE DISTÂNCIAS

Para agrupar dados, é necessária uma forma de medir os elementos e suas distâncias relativas, com o intuito de decidir quais elementos pertencem a um grupo. Essa distância pode representar uma semelhança, uma dissemelhança ou uma "força"entre os elementos. Como apresentado em (JAIN; MURTY; FLYNN, 1999), o processo de agrupamento de elementos utiliza como critério sua similaridade ou dissimilaridade, de forma que os grupos formados se apresentem homogêneos internamente e heterogêneos externamente.

Para isso, uma matriz é calculada e cada elemento dessa matriz representa uma medida de distância entre dois elementos. Essa matriz é chamada de matriz de distâncias. Tal matriz é quadrada, possuindo dimensão *n* x *n*, com *n* representando o número de elementos envolvidos. Sua diagonal principal é nula pois ela representa a distância de um elemento para ele mesmo. Trata-se de uma matriz simétrica. Neste trabalho, utilizou-se a porção triangular inferior da matriz. Para melhor exemplificar, a Tabela 2 apresenta uma matriz de distâncias obtida a partir dos dados apresentados na Tabela 1.

	1	**2**	**3**	**4**
1	0,00	-	-	-
2	5,83	0,00	-	-
3	9,64	7,55	0,00	-
4	4,69	7,35	8,06	0,00

Tabela 2: Matriz de distâncias referente aos dados da Tabela 1.

Para agrupamentos hierárquicos, como medidas de distância, tem-se por exemplo a distância Euclidiana, a distância Euclidiana quadrada, a distância de Manhattan e a distância de Chebychev, entre outras (DASH et al., 2003). Nesse trabalho será utilizada a distância Euclidiana, por ser, de acordo com (BOUGUETTAYA et al., 2015), um dos métodos de se calcular distâncias mais usual em algoritmos hierárquicos. Tal medida é a distância geométrica no espaço multidimensional. O cálculo da distância Euclidiana entre dois elementos $\boldsymbol{X} = [\boldsymbol{X_1}, \boldsymbol{X_2}, \ldots, \boldsymbol{X_n}]$ e $\boldsymbol{Y} = [\boldsymbol{Y_1}, \boldsymbol{Y_2}, \ldots, \boldsymbol{Y_n}]$ no $\boldsymbol{R_n}$, é definido na fórmula 2.1.

$$d_x y = \sqrt{(X_1 - Y_1)^2 + (X_2 - Y_2)^2 + \ldots + (X_n - Y_n)^2} = \sqrt{\sum_{i=1}^{N}(X_1 - Y_1)^2} \qquad (2.1)$$

A seção a seguir apresenta o conceito do coeficiente de correlação cofenética e sua utilização na avaliação de dendrogramas obtidos por clusterização hierárquica.

COEFICIENTE DE CORRELAÇÃO COFENÉTICA

Em algoritmos de clusterização hierárquica, o coeficiente que estabelece a correlação entre os elementos da matriz de distâncias original, conhecida como matriz fenética, e a matriz construída após o processo de agrupamento, conhecida como matriz cofenética, é usualmente considerado como uma medida de avaliação da consistência do agrupamento (FILHO; RIBEIRO; BURIN, 2010). Tal coeficiente é conhecido como coeficiente de correlação cofenética (*ccc*) e foi proposto por (SOKAL; ROHLF, 1962). Ele é usado como uma medida de consistência da matriz de distâncias para o seu correspondente agrupamento. Isso quer dizer que tal coeficiente permite medir o grau de associação linear entre as duas matrizes supra citadas (fenética e cofenética).

Alguns trabalhos na literatura, como (EVERITT, 1979) e (JAIN; MURTY; FLYNN, 1999), afirmam que valores para tal coeficiente acima de 0,7 são indicativos de um bom dendrograma, a partir do qual pode ser extraída a estrutura natural dos clusters presentes no conjunto de dados. Contudo, tal valor não é um consenso, visto que posteriormente tais dendrogramas precisam ser interpretados. Outro fator que deve ser levado em consideração é que nem sempre é possível alcançar valores muito próximos de um a partir de determinados agrupamentos.

Para a construção da matriz cofenética, é preciso substituir os valores da matriz de distâncias, utilizada como entrada de dados, pelos valores correspondentes à distância onde ocorreram as junções entre os elementos. Conforme demonstrado na Tabela 4, a partir dos dados da Tabela 3, os valores onde ocorrem as junções estão presentes na matriz fenética.

	1	**2**	**3**
1	5,83	-	-
2	9,64	7,55	-
3	4,69	7,35	8,06

Tabela 3: Matriz fenética.

	1	**2**	**3**
1	5,83	-	-
2	7,55	7,55	-
3	4,69	5,83	7,54

Tabela 4: Matriz cofenética.

Os valores do *ccc* podem variar no intervalo de 0 a 1, sendo que quanto mais próximo de 1, mais consistente é a matriz cofenética utilizada no cálculo. E por consequência, o

dendrograma gerado a partir de tal matriz apresenta um menor grau de distorção, sendo assim mais representativo. Esse coeficiente é calculado utilizando a fórmula 2.2.

$$c = \left(\frac{\sum_{i<j} (x(i,j) - x)\ (t(i,j) - t)}{\sqrt{\left[\sum_{i<j} (x(i,j) - x)^2\right] \left[\sum_{i<j} (t(i,j) - t)^2\right]}} \right) \tag{2.2}$$

Suponha que os dados originais, aqui representados pelo grupo *Xi*, foram agrupados hieraquicamente, considerando a menor distância entre dois pontos, produzindo um dendrograma. Na fórmula 2.1, $x(i, j) = | X_i \neq X_j |$ é a distância euclidiana entre o i-ésimo e j-ésimo ponto da matriz fenética. Já *t (i, j)* é a distância dendogramática (cofenética) entre o i-ésimo e j-ésimo ponto. Tal distância representa a altura onde o agrupamento foi gerado. Ainda tem-se *x* sendo a média dos valores da matriz fenética e *t* sendo a média dos valores da matriz cofenética.

Como pode ser notado no assunto abordado nessa seção, percebe-se que o coeficiente de correlação cofenética é um importante instrumento que nos permite analisar a qualidade de agrupamentos hierárquicos gerados.

Na seção a seguir, é explicado como funciona um algoritmo de clusterização hierárquica. Também é mostrado, passo-a-passo, um exemplo do funcionamento do método *single linkage*, que é a heurística subordinada utilizada nesse trabalho.

MÉTODOS AGLOMERATIVOS

Na clusterização hierárquica aglomerativa, cada elemento inicia-se representando um grupo, e a cada passo, um elemento é ligado a outro de acordo com sua similaridade, até o último passo, onde é formado um grupo único com todos os elementos. Existe uma variedade de métodos aglomerativos, que são caracterizados de acordo com o critério utilizado para definir as distâncias entre os grupos. De acordo com (ANDERBERG, 2014), a maioria dos métodos parecem ser formulações alternativas de três grandes conceitos de agrupamento aglomerativo: métodos de ligação (*single linkage, complete linkage, average linkage, median linkage*), métodos centróide e métodos de soma de erros quadráticos ou variância (método Ward). Os métodos aglomerativos, de forma geral, utilizam os passos de um algoritmo padrão, conforme descrito na Figura 2. A diferença entre os métodos ocorre no passo 4, onde a função distância é definida de acordo com cada método (JOHNSON; WICHERN et al., 1992).

Entrada: base de dados com *n* elementos
1. Iniciar com *N* grupos, contendo um elemento cada grupo e uma matriz de similaridade
2. Repetir
3. Localizar maior similaridade
4. Atualizar a matriz de similaridade, retirando os elementos de menor distância e atualizando as distâncias dos elementos restantes para o grupo formado
5. Até *n-1*, quando todos os elementos estarão em um único grupo

Figura 2: Algoritmo utilizado pelos métodos de clusterização hierárquica.

Nesse trabalho será utilizado o método de agrupamento *single linkage*, que emprega a distância de valor mínimo, ou seja, a menor distância entre os elementos para formar um grupo. Nesse método, a similaridade entre dois *clusters* é a menor distância entre os pontos nos dois *clusters* (DASH et al., 2003). De acordo com (SRINIVAS; MOHAN, 2010), tal algoritmo de clusterização tem sido utilizado em várias aplicações e é considerado como mais versátil que outros algoritmos de agrupamento. Devido a isso, este método foi utilizado nesse trabalho.

A seguir, para se ter uma melhor compreensão de como o método *single linkage* funciona, tem-se os passos para obtenção do dendrograma da Figura 1 utilizando os dados da Tabela 1. Inicia-se o processo conforme o passo 1 da Figura 2, especificado na Tabela 2. Ou seja, uma matriz de distâncias é utilizada como entrada para o algoritmo. Após, realiza-se o passo 3 da Figura 2 conforme especificado na Tabela 5 em destaque, fazendo a junção do elemento 1 com o 4.

	1	**2**	**3**	**4**
1	0,00	-	-	-
2	5,83	0,00	-	-
3	9,64	7,55	0,00	-
4	4,69	7,35	8,06	0,00

Tabela 5: Matriz inical - Localizando a menor distância.

Continuando o processo, realiza-se o passo 4 da Figura 2, que é a atualização da matriz de similaridade, tendo a Tabela 6 como resultado.

	14	**2**	**3**
14	0,00	-	-
2	5,83	0,00	-
3	8,06	7,55	0,00

Tabela 6: Formado primeiro cluster e tabela atualizada.

Novamente, executa-se o passo 3 na Tabela 6 em destaque, fazendo a junção do

grupo com os elementos 1 e 4 com o elemento 2, que é a menor distância apresentada na tabela. Posteriormente, a matriz de similaridade é atualizada, e tem-se a Tabela 7 como resultado.

Como foram executados n ≠ 1 passos, falta juntar o último elemento (3), fazendo a junção do grupo com os elementos 1, 4 e 2 com o elemento 3 e aciona-se o critério de parada. Um detalhe importante, é que as menores distâncias selecionadas (em destaque nas Tabelas 5 , 6 e 7) serão utilizadas para compor a matriz cofenética, conforme apresentada na Tabela 4.

	(14)2	**3**
(14)2	0,00	-
3	7,55	0,00

Tabela 7: Situação final da matriz.

Nessa seção foi apresentado o conceito de métodos aglomerativos, utilizados na clusterização hierárquica. Também foi exposto um exemplo do funcionamento do método *single linkage*. Esse entendimento facilitará a compreensão do que é proposto nesse trabalho.

Na seção a seguir, são apresentados alguns trabalhos que também fizeram uso de métodos aglomerativos para resolução de seus problemas. O objetivo de se apresentar esses estudos é mostrar a grande aplicabilidade prática de tais métodos apresentam.

APLICAÇÕES PRÁTICAS DA CLUSTERIZAÇÃO HIERÁRQUICA

O problema da clusterização possui muitas aplicações nas mais variadas áreas de pesquisa como bioinformática, (HOUTE; HERINGA, 2010) e (ANDREOPOULOS et al., 2009), redes de comunicações (BANDYOPADHYAY; COYLE, 2003), mineração de dados (NG; HAN, 2002), entre outras. Com o intuito de exemplificar a variedeade de situações e problemas em que esta técnica pode ser empregada, nesta seção, apresentaremos algumas de suas aplicações.

Clusterização para formação de Células de Manufatura (PFCM)

De acordo com (TRINDADE, 2004), existem industrias que produzem uma pequena variedade de produtos e um alto volume de produção. Elas geralmente organizam o ambiente de produção em linhas de produção, sendo que cada linha de produção é composta de vários tipos de máquinas, dedicadas exclusivamente à produção de um único produto.

Porém, em outras industrias é produzido uma grande variedade de produtos, com um volume médio de produção de cada item. Portanto, elas precisam de um modelo diferente de produção. Uma abordagem diferente é a formação de grupos (*clusters*) de máquinas com funcionalidades idênticas, formando departamentos especializados em uma função específica. Assim, uma parte de um produto que necessite de operações de manufatura de mais de um tipo de máquina, precisará percorrer todos os grupos que contenham os tipos de máquinas necessários para a sua formação.

Nos últimos anos um novo modelo de organização destes sistemas vem sendo usado, denominado manufatura celular. A manufatura celular (TRINDADE; OCHI, 2004) consiste em agrupar máquinas de diferentes funcionalidades para a manufatura de produtos com diferentes partes. O sistema de produção fica dividido em vários grupos ou *clusters* formados por células de máquinas e famílias de partes, denominados células de manufatura.

As Tabelas 8 e 9 mostram a formação de células de manufatura. As linhas (M1, M2, M3, M4, M5 e M6) especificam as máquinas e as colunas (P1, P2, P3, P4, P5, P6, P7 eP8) especificam as partes dessas máquinas. Nesse caso, se deseja formar três grupos, com no máximo 2 máqunas por cluster e no máximo 3 partes por *cluster*.

	P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
M1	1			1			1	
M2		1			1			1
M3	1		1	1			1	
M4	1	1			1			1
M5			1			1		
M6			1			1		

Tabela 8: Matriz de entrada para o PFCM.

	P1	**P4**	**P7**	**P3**	**P6**	**P2**	**P5**	**P8**
M1	1	1	1					
M3	1	1	1	1				
M5				1	1			
M6				1	1			
M2						1	1	1
M4	1					1	1	1

Tabela 9: Agrupamentos formados para PFCM.

A formação de *clusters* seguindo este modelo resulta em diversas vantagens para a gestão da produção, tais como:

- Redução do transporte de material de produção;
- Redução do tempo de produção dos produtos e consequente aumento da capacidade de produção;
- Simplificação do gerenciamento e controle do sistema de produção, agora divididos em vários subsistemas independentes;
- Simplificação do escalonamento das atividades, que agora será feito separadamente em cada *cluster*;

Para maiores informações sobre este problema, consulte (TRINDADE, 2004).

Clusterização aplicada ao Mercado Financeiro Brasileiro

Evidências indicam que ações de empresas de um mesmo setor da economia apresentam retornos semelhantes ao longo do tempo, uma vez que estariam expostas a variáveis econômico-financeiras e técnico-operacionais semelhantes. Gestores de recursos, de maneira geral, utilizam esta evidência em suas avaliações diárias na busca pelos melhores investimentos. Entretanto, na grande maioria dos casos, não há um embasamento teórico e matemático que comprove essa relação entre as ações.

Para verificar tal relação, foi averiguado se, para um grupo de ações classificadas como mais relevantes dentre as presentes na Bolsa de Valores brasileira, os preços diários de fechamento, que se comportam analogamente, correspondem a empresas de um mesmo setor econômico. Para testar tal hipótese, foram avaliados diferentes métodos de clusterização aplicados a matriz de dissimilaridade entre os dados estudados.

A Figura 3 apresenta um resumo do resultado desta clusterização. Cada coluna representa um cluster que foi formado. As diferentes cores identificam o ramo de atuação de cada ação. Como pode ser verificado, foram formados 10 *clusters*, sendo que os maiores foram os clusters 2, 4 e 5. Os clusters formados foram agrupados de acordo com o rendimento das ações observados em um determinado intervalo de tempo.

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9	Cluster 10
GETI4	AMBV4	NATU3	BVMF3	BBRK3	BRML3	CMIG4	CSMG3	EQTL3	ESTC3
TBLE3	BRFS3	PCAR4	EMBR3	BISA3	IGTA6	CESP6			KROT11
	CSAN3	RADL3	OHLB3	CYRE5	MULT3	CPLE6			
	AMIL3	CRUZ3	LPSB3	EVEN5		CPFL3			
	DASA3	HYPE3	TIMP3	EZTC3		ELET6			
	MPXE3	BTOW3	OIBR4	GFSA3		ELPL4			
	GUAR3	LAME4	GGBR4	HBOR5		ENBR3			
	KLBN4	HGTX3	CSNA3	MRVE3		LIGT3			
	MYPK3	AMAR3	USIM5	PDGR3		SBSP3			
	POMO4	LREN3	MMXM3	RSID3		TRPL4			
	RAPT4		VALE5	TCSA3					
	WEGE3		SUZB5	GOLL4					
	LLXL3		BRKM5	CCRO3					
	STBP11		MAGG3	VAGR3					
	VIVT4		MRFG3	OGXP3					
	BBDC4		JBSS3	DTEX5					
	BBAS3		MDIA3	RENT3					
	ITUB4		ODPV3	TOTS3					
	PSSA3		PETR4						
	SULA11								

Setor
Petróleo, Gás e Biocombustíveis
Materiais Básicos
Bens Industriais
Construção e Transporte
Consumo Não-cíclico
Consumo Cíclico
Tecnologia da Informação
Telecomunicações
Utilidade Pública
Financeiros e Outros

Figura 3: Resumo do resultado da clusterização em ações do mercado financeiro.

Como pode ser verificado na Figura 4, alguns clusters realmente foram formados por ações que atuam no mesmo setor da economia. Entretanto, tais clusters eram os menores. Contudo, em outros clusters (os maiores) que apresentaram os mesmo rendimento em um determinado intervalo de tempo, houve uma variação dos setores da economia em que as ações daquele grupo atuavam. A hipótese levantada pelo autor do estudo se mostrou parcialmente verdadeira. Mas o que vale destacar aqui é a aplicabilidade das técnicas de agrupamento no problema em questão. Para maiores informações sobre esse problema, consulte (TORRES, 2013).

Clusterização aplicada à bioinformática

Aberrações cromossômicas tendem a ser características para dados (sub)tipos de câncer. Tais aberrações tendem a ser detectadas com uma matriz de hibridização genômica comparativa (aCGH). Agrupamentos de perfis de tumores aCGH auxiliam na identificação de regiões cromossômicas de interesse e fornece informações de diagnóstico úteis sobre o tipo de câncer. Um aspecto importante é que a medida de perfis tumorais individuais aCGH pode ser atribuída a agrupamentos associados a um dado tipo de câncer.

Para realização de tal processo de agrupamento, foi utilizado um novo algoritmo de agrupamento *fuzzy* evolucionário (EFC), que é adequado para lidar com agrupamentos

sobrepostos. Esse método avalia estas sobreposições usando graus de adesão de *cluster*, que são usados como uma medida de confiança para cada amostra, a ser atribuído a um determinado tipo de tumor.

Como exemplo, na Figura 4 é mostrado como o EFC revela a sobreposição de perfis de tumor de agrupmanentos aCGH. Graus de adesão são visualizadas como um tetraedro, onde as distâncias das amostras para os cantos são inversamente proporcionais aos graus de pertinência correspondentes. Uma sobreposição pode ser diferenciada entre os agrupamentos de cólon e gástrico, nos quais sete amostras são envolvidas. Além disso, numerosas amostras de linfoma mostram um (relativo) elevado grau de adesão para agrupamentos de cólon e/ ou gástricos, enquanto as amostras de cabeça e pescoço são bem separados.

A Figura 4A e a Figura 4B revelam conjuntos de perfis sobrepostos aCGH tumorais. Graus de adesão são visualizados como um tetraedro. Linfoma está destacado na cor amarela; cabeça e pescoço, na cor vermelha; gástrico, na cor lilás; e cólon, na cor verde. A Figura 4A apresenta uma visão lateral. E a Figura 5B apresenta uma vistão superior.

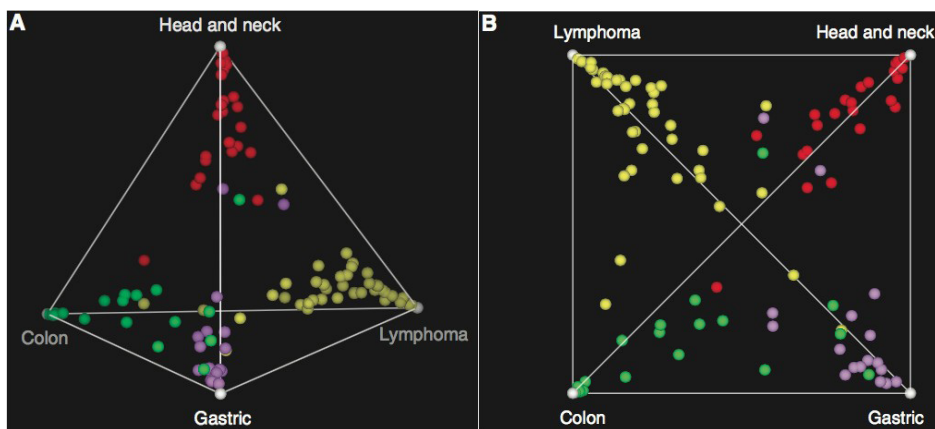



Figura 4: Visão lateral (A) e visão superior (B) da sobreposição de clusters.

Como pode ser verificado nas figuras 5A e a 5B foi possível utilizar um método específico de clusterização para se trabalhar com grupos sobrepostos. O interessante desse trabalho e a identificação dos clusters em uma figura geométrica tridimensional. Outra vez, percebe-se o quão vasto são as areas de aplicação das técnicas de clusterização.Para maiores informações sobre esse problema, consulte (HOUTE; HERINGA, 2010).

Nessa seção foram apresentados alguns estudos onde as técnicas de clusterização se mostraram de grande valia para a resolução de problemas em áreas de conhecimento bem distintas. Isso mostra o grande potencial de aplicação de tais técnicas bem como os bons resultados que elas podem proporcionar.

Na seção a seguir será apresentada e descrita a metaheurística GRASP. Também nessa seção, será explicado como tal técnica foi utilizada para melhorar o ccc, utilizando a heurística *single linkage* como método construtivo.

HEURÍSTICA GRASP PROPOSTA

De acordo com (OCHI; DIAS; SOARES, 2004), no processo de clusterização, a busca pela melhor solução no espaço de soluções viáveis é um problema NP-Completo, pois a avaliação exaustiva de todas as configurações de clusterizações possíveis é computacionalmente inviável, restringindo com isso o uso de métodos exatos para a sua solução. Devido a isso, métodos heurísticos tem sido propostos com frequencia, os quais fornecem soluções sub-ótimas com significativa redução da complexidade na solução do problema.

Contudo, as heurísticas são métodos de busca que tiram proveito de características e informações do próprio problema a ser explorado, facilitando o encontro de um mínimo global no espaço de busca (BLUM; ROLI, 2003). As heurísticas são limitadas e fornecem sempre a mesma solução quando iniciadas de um mesmo ponto de partida. As metaheurísticas vêm suprir esta limitação e objetiva explorar um espaço de pesquisa de forma inteligente, isto é, encontrar soluções de alta qualidade movendo-se para áreas não exploradas quando necessário.

A metaheurística GRASP (*Greedy Randomized Adaptative Search Procedures*) que significa em português Procedimentos de Busca Adaptativos Gulosos Randomizados, foi desenvolvida por (FEO; RESENDE, 1995) como uma tentativa de obter bons resultados para problemas difíceis de otimização combinatória. Atualmente esta metaheurística de construção e melhoria possui um grande destaque na literatura devido aos bons resultados obtidos nas aplicações em problemas de otimização combinatória, pela facilidade de paralelização e principalmente por sua flexibilidade (FESTA; RESENDE, 2009).

O GRASP é basicamente um procedimento iterativo, onde cada iteração consiste de duas fases:

- uma **fase de construção** de uma solução inicial de forma gulosa, aleatória e adaptativa;
- uma **fase de busca local**, que objetiva melhorar a solução obtida, explorando sua vizinhança;

A melhor solução dentre todas as iterações será então o resultado final (FEO; RESENDE, 1995). Na Figura 5 é apresentado o pseudocódigo do algoritmo GRASP.

Dada a característica gulosa ingênua do método *single linkage* para clusterização hierárquica, resolveu-se nesse trabalho fazer uso da metaheurística GRASP que faz bom uso dessa característica gulosa para obtenção de melhores soluções para o problema em questão.

$f* \leftarrow \infty$;

Para l *iteracões* **Faça**

$x \leftarrow GulosoAleatorizado(\ )$;

$x \leftarrow BuscaLocal(x)$;

Se $f(x)$ *menor* $f*$ **Então**

$f* \leftarrow f(x)$;

$x* \leftarrow x$;

Fim Se

Fim Para

Retornar $x*$;

Fim GRASP

Figura 5: Algoritmo GRASP.

Conforme pode ser verificado na Figura 5, as iterações da metaheurística GRASP envolvem uma heurística subordinada aleatorizada seguida de uma busca local. Dessa maneira, espera-se obter melhores ccc do que os obtidos pelo método *single linkage*.

A próxima seção tratará da fase de construção da metaheurística GRASP e também será explicado como a mesma foi adequada para o problema abordado por esse trabalho.

FASE DE CONSTRUÇÃO

Na fase de construção, a solução viável é iterativamente construída elemento por elemento. A heurística é adaptativa porque os benefícios associados com cada elemento são atualizados a cada iteração da fase de construção para refletir as mudanças ocorridas pela seleção de elementos anteriores. A parte aleatória corresponde à forma de escolha dos melhores candidatos da lista. De acordo com (MORELLI; VIEIRA, 1999), cada iteração é composta por três etapas:

- Construção da Lista Restrita de Candidatos (LRC), a qual contém um conjunto reduzido de elementos candidatos a pertencer à solução;
- Escolha aleatória de um elemento da LRC e inclusão de elemento na solução;
- Adaptação ou recálculo da função gulosa para os elementos ainda não pertencentes à solução. Ressalta-se que o tamanho da LRC é controlado por um parâmetro –, onde para – = 0 tem-se um comportamento puramente guloso do algoritmo e para – = 1, um comportamento aleatório. A componente probabilística do método é devida à escolha aleatória de um elemento da LRC. Este procedimento permite que diferentes soluções de boa qualidade sejam geradas.

Nesse trabalho, as 3 etapas da fase construtiva ocorrem da seguinte maneira:

- Etapa 1: a LRC é formada pelos pares de dados a serem agrupados naquele passo, que são arrumados em ordem crescente de distância entre os mesmos.
- Etapa 2: Escolhe-se aleatoriamente um elemento da LRC que possua distância entre os pares de objetos que esteja entre min e min + –(max - min), onde min e max são respectivamente o menor e o maior valor de distância.
- Etapa 3: Recalcula-se as distâncias entre os elementos no grupamento, considerando agora, os elementos agrupados na Etapa 2.

A forma que o GRASP implementa suas características adaptativas e aleatórias foram vistas nessa seção. Na próxima seção, será abordada a outra fase do GRASP, que é a busca local, onde se espera, através da exploração da vizinhança, melhorar o resultado já obtido na fase de construção.

FASE DE BUSCA LOCAL

Métodos de busca local em problemas de otimização constituem uma família de técnicas baseadas na noção de vizinhança, ou seja, são métodos que percorrem o espaço de pesquisa passando, iterativamente, de uma solução para outra que seja sua vizinha (ALVARENGA; ROCHA, 2006).

A fase de busca local de GRASP aproveita a solução inicial da fase de construção e explora a vizinhança ao redor desta solução. Se um melhoramento é encontrado, a solução corrente é atualizada e novamente a vizinhança ao redor da nova solução é pesquisada. O processo se repete até nenhum melhoramento ser encontrado. É preciso ter cuidado em:

- escolher uma vizinhança apropriada;
- usar estruturas de dados eficientes para acelerar a busca local;
- ter uma boa solução inicial;

Neste trabalho, a cada iteração, busca-se fazer a troca de dois elementos de posição na árvore hierárquica e recalcula-se o *ccc*. Caso essa troca proporcione um melhor valor de *ccc* que o melhor até o momento, essa troca é efetivada e essa passa a ser a solução corrente. Após a execução de todas as iterações, a busca local retorna a solução e o melhor valor do coeficiente calculado. Dessa forma, a busca local tenta melhorar, a cada iteração, o *ccc* que foi obtido na fase de construção.

A próxima seção tratará sobre a programação paralela. Esse tipo de programação foi utilizado nesse trabalho para implementar um dos algoritmos desenvolvidos. Tal solução apresenta como grande benefício a possibilidade de se utilizar matrizes de grandes dimensões como entrada de dados. Algo que a solução GRASP sequencial não é capaz de fazer, devido a suas limitações de processamento e memória.

PROGRAMAÇÃO PARALELA

De acordo com (IGNÁCIO; FILHO, 2002), muitos problemas de otimização não podem ser resolvidos de forma exata, utilizando a computação convencional (sequencial) dentro de um tempo razoável, o que inviabiliza sua utilização em muitas aplicações reais. Embora os computadores estejam cada vez mais velozes, existem limites físicos e a veloci- dade dos circuitos não pode continuar melhorando indefinidamente. Por outro lado, nos últimos anos tem-se observado uma crescente aceitação e uso de implementações paralelas nas aplicações de alto desempenho como também nas de propósito geral, motivados pelo surgimento de novas arquiteturas que integram dezenas de processadores rápidos e de baixo custo. Essas arquiteturas compõem ambientes com memória distribuída como, por exemplo, estações de trabalho ou PCs conectados em rede.

Para melhor entender a concepção das diferentes arquiteturas paralelas, pode-se utilizar a técnica de classificação de Flynn (WILKINSON; ALLEN, 1999), a qual é amplamente adotada. Esta classificação está baseada na relação que existe entre o fluxo de instruções e o fluxo de dados. Neste sentido, quatro classes podem existir:

- *Single Instruction Single Data* (SISD) - Onde para cada instrução há um único dado sendo operado. Geralmente computadores pessoais possuem essa configuração.
- *Multiple Instruction Single Data* (MISD) - Neste caso, hipoteticamente haveria vá- rias instruções sendo operadas sobre o mesmo dado.
- *Single Instruction Multiple Data* (SIMD) - São as máquinas vetoriais, ou seja, uma única instrução é executada sobre múltiplos dados de forma paralela.
- *Multiple Instruction Multiple Data* (MIMD) - Aqui mútiplas instruções são executadas sobre múltiplos dados. Tais execuções paralelas podem ocorrer através de multiprocessadores ou multicomputadores.

De acordo com essa classificação, os computadores paralelos podem ser classificados em SIMD e MIND. As máquinas SIMD executam uma instrução de cada vez, sobre diversos conjuntos de dados. Nessa categoria estão as máquinas vetoriais e matriciais. As máquinas MIMD rodam programas diferentes, em processadores diferentes, e podem ser divididas em multiprocessadores que compartilham a memória principal e os multicomputadores que não compartilham nenhuma memória. Segundo (TANENBAUM, 2006), os multicomputadores podem ser divididos em máquinas MPPs (*Massive Parallel Processor*) e COWs (*Cluster of Workstations*). Os MPPs são os supercomputadores que utilizam processadores padrão como o IBM RS/6000, a família Dec Alpha ou a linha Sun UltraSPARC. Uma outra característica dos MPPs é o uso de redes de interconexão

proprietárias, de alta performance, baixa latência e banda passante alta.

Um COW é composto de vários computadores pessoais ou estações de trabalho, ligados por meio uma rede de computadores comercial. Esse tipo de ambiente já é bastante utilizado em empresas e instituições de ensino, principalmente devido ao fato de ser mais acessível que outros computadores paralelos de alto custo. Como exemplo, pode-se citar o sistema COW chamado Galaxi, apresentado por (DENG; KOROBKA, 2001), e implementado sobre uma rede Fast Ethernet e Gigabit Ethernet.

Entretanto, além da comunicação em rede, também se faz necessária uma camada de software que possa gerenciar o uso dessas máquinas de forma paralela. E para isso, existem bibliotecas especializadas no tratamento da comunicação entre processos e na sincronização de processos concorrentes (IGNACIO; FILHO; GALVÃO, 2000). Geralmente, tais bibliotecas podem ser usadas tanto por máquinas classificadas como MPP como por máquinas classificadas como COW, com a possibilidade ainda de poder realizar a transferência de aplicações entre as plataformas. Em ambiente de multicomputadores, ainda de acordo com (IGNACIO; FILHO; GALVÃO, 2000), os dois sistemas mais utilizados são o MPI (*Message-Passing Interface*) e o PVM (*Parallel Virtual Machine*). Ambos os sistemas se baseiam nas trocas de mensagens para realizar o paralelismo.

PVM é um pacote de software que permite que um ambiente computacional heterogêneo (COW) seja programado como se fosse uma única máquina paralela virtual (GEIST, 1994). Contudo, já existem modificações que permitem que esse pacote rode em máquinas MPP (BEGUELIN et al., 1991). A implementação do PVM é baseada em processos do Unix, onde cada tarefa PVM é um processo Unix. Isto explica parcialmente a alta portabilidade do sistema para computadores de arquiteturas tão diferentes.

MPI é um padrão de interface para a troca de mensagens em máquinas paralelas com memória distribuída (GROPP; LUSK; SKJELLUM, 1999). O MPI oferece mais opções e mais parâmetros por chamada, e vem se tornando o padrão das implementações paralelas tanto em ambientes MPP como em ambientes COW (IGNÁCIO; FILHO, 2002). Por apresentar tais características, o MPI foi escolhido para a implementação da solução paralela e distribuída da metaheurística GRASP aplicada ao problema de clusterização, visando ganho de desempenho.

Existem diversas implementações do padrão MPI para as mais diversas linguagens. Nesse trabalho foi utilizado, especificamente, uma implementação desse padrão feita em Java, chamada MPJ Express (SHAFI; MANZOOR, 2009), em especial, devido ao conhecimento do autor a respeito da instalação e configuração dessa ferramenta.

Nessa seção foi apresentado os principais conceitos relacionados a programação

paralela e seus principais padrões. Também foi explicado por que o padrão MPI foi escolhido para ser utilizado nesse trabalho. A próxima seção trás uma breve detalhamento sobre as características do MPI.

O PADRÃO MPI

O MPI é um padrão que especifica um modelo (um conjunto de interfaces) e não sua implementação. O modelo de programação de passagem de mensagem é bastante poderoso por ser fácil de usar e possui alta portabilidade (BARKER, 2015). A facilidade de uso se justifica tanto pelo uso de interfaces simples, claramente especificadas, quanto ao seu comportamento.

A portabilidade é obtida, pelo fato de que com o uso de passagem de mensagem, é possível escrever programas que irão funcionar em máquinas dos mais variados tipos, como multiprocessadores com memória distribuída, redes de estações de trabalho ou ambientes híbridos dos mais variados tipos.

A ideia geral é permitir escrever aplicações que usem passagem de mensagem como modelo de programação de forma padronizada possibilitando interoperabilidade entre os programas e consequente aumento de escalabilidade. Atualmente, o padrão MPI é utilizado principalmente em máquinas paralelas e clusters (conjuntos) de estações de trabalho (EL-REWINI; ABD-EL-BARR, 2005).

Na próxima seção será abordado os conceitos básicos de processamento paralelo. Também na próxima seção, será apresentada e discutida as principais funções do padrão MPI.

CONCEITOS BÁSICOS DE PROCESSAMENTO PARALELO

Para compreender como ocorre a comunicação ponto a ponto em um ambiente de paralelismo, o entedimento de alguns conceitos se fazem necessários:

- Processo, nesse contexto, é um módulo executável identificado por um rank, que é algo similar ao pid (*process id*) em sistemas operacionais baseados em Unix. Seu valor pode variar de 0 a N-1 (onde N é o número total de processos do ambiente paralelo/distribuído).
- Grupo pode ser entendido como uma coleção ordenada de processos.
- Contexto é uma entidade que serve para separar o espaço de comunicação. Ou seja, uma mensagem pertencente a um contexto não pode interferir em outro contexto.
- Comunicador é um conceito que encapsula o conceito de grupo e contexto, e de- fine como será a interação entre os processos. O comunicador mais comum

é o MPI_COMM_WORLD, que permite a colaboração entre processos do ambiente paralelo/distribuído. Os comunicadores ainda podem ser classificados em intracomuni- cadores, que define a comunicação dentro do grupo de processos, e intercomunicadores, que permite a comunicação entre grupos distintos de processos.

No padrão MPI, uma aplicação é constituída por um ou mais processos que se comunicam, acionando-se funções para o envio e recebimento de mensagens entre os processos. Essas operações podem ser síncronas (esperam a operação completar antes de prosseguir com a execução) ou assíncronas (não esperam o término da operação para prosseguir). Inicialmente, na maioria das implementações, um conjunto fixo de processos é criado. E esses processos podem executar diferentes programas.

O MPI possui um grande conjunto de funções. Contudo, uma grande parte das aplicações que utilizam esse padrão fazem uso de praticamente apenas 6 funções:

- MPI_INIT - Inicia a execução MPI
- MPI_FINALIZE - Finaliza a execução MPI
- MPI_COMM_SIZE - Determina o número de processos
- MPI_COMM_RANK - Determina a identificação de processos
- MPI_SEND - Função utilizada para o envio de mensagens
- MPI_RECV - Função utilizada para o recebimento de mensagens

Agora que já discorrido sobre os conceitos básicos de processamento paralelo e apresentada as principais funções do MPI, na próxima seção trataremos das funções utilizadas em operações coletivas. A execução de processos em grupo foi uma técnica utilizada nesse trabalho na implementação da solução GRASP paralela.

OPERAÇÕES COLETIVAS NO PROCESSAMENTO PARALELO

Frequentemente, algoritmos paralelos necessitam de operações coordenadas envolvendo múltiplos processos. E para isso, o padrão MPI oferece um conjunto de funções especializadas para esse tipo de algoritmo. Tais funções são rotinas coletivas de movimentação de dados, nas quais todos os processos interagem com origens distintas para, por exemplo, espalhar, juntar e distribuir os dados. Algumas dessas funções são apresentadas a seguir.

- MPI_BARRIER (Barreira) - utilizada para sincronizar a execução de processos de grupo. Nenhum processo pode realizar qualquer instrução, até que todos os processos tenham passado por essa barreira.
- MPI_BCAST(Difusão) - implementa um processo de dispersão de dados do tipo

um para todos, no qual um único processo origem envia um dado para todos os outros processos e cada processo recebe esse dado.

- MPI_GATHER(Juntar) - implementa um processo de junção dos dados de todos os processos em um único processo.
- MPI_SCATTER(Espalhar) - implementa uma operação de dispersão de um conjunto de dados para todos os processos.
- MPI_REDUCE/MPI_ALLREDUCE - implementam operações de redução, combinando valores do *buffer* de entrada de cada processo e retornando um valor combinado para o *buffer* de saída de um único processo de origem.

Desse conjunto de funções, na implementação realizada nesse trabalho, apenas a MPI_BARRIER foi utilizada, pois a mesma permite que conjunto de iterações de um algoritmo seja distribuído entre as máquinas que compõem um cluster. Utilizando essa função, o algoritmo só é finalizado quando o nó do cluster que iniciou a execução recebe a resposta de todos os processos enviados para os demais nós. Assim, cada máquina retorna seu melhor *ccc* e apenas o melhor de todos será considerado como solução.

A próxima seção explana brevemente sobre quais as etapas necessárias para executar um algoritmo em um ambiente de processamento paralelo e distribuído COW.

ETAPAS PARA EXECUÇÃO EM UM AMBIENTE DE PROCESSAMENTO PARALELO

Quando se implementa um programa que utiliza as funções do MPI, é necessário realizar a sua compilação e execução. Nesse ponto, vale ressaltar que a forma com que a implementação do MPI é realizada, vai influenciar em como serão os métodos de compilação e execução. Contudo, se for feita uma análise de forma bem abrangente, as principais etapas para a compilação e execução de um programa MPI são:

- Utilizar compiladores que reconheçam os métodos MPI, ou incluir bibliotecas MPI no processo de compilação;
- Verificar no ambiente de computação paralela, quais são os nós (computadores em rede) disponíveis.
- Definir o protocolo de comunicação a ser utilizado;
- Enviar uma cópia do programa para cada nó que será utilizado no cluster de computadores;
- Executar o programa;

Agora que já se conhece os passo necessários para compilar e executar um programa que utiliza o padrão MPI, já se pode abordar a implementação java de tal padrão que foi utilizada nesse trabalho. Esse é o assunto da próxima seção.

MPJ EXPRESS

MPJ Express implementa, em alto nível, as funcionalidades do MPI em java puro, o que permite aos desenvolvedores escrever e executar aplicações paralelas para processadores multicore e clusters computacionais (TOMAR et al., 2013). O MPJ Express é implementado em camadas, o que possibilita o desenvolvimento incremental, e provê a capacidade de atualizar e alterar camadas, modificando sua estrutura quando necessário (BAKER; CARPENTER; SHAFT, 2006). A Figura 6, obtida em http://mpj-express. org/ software/mpj-design-newest.png, mostra uma visão em camadas do sistema de mensagens do MPJ, desde o seu nível mais alto até o seu sistema mais básico.

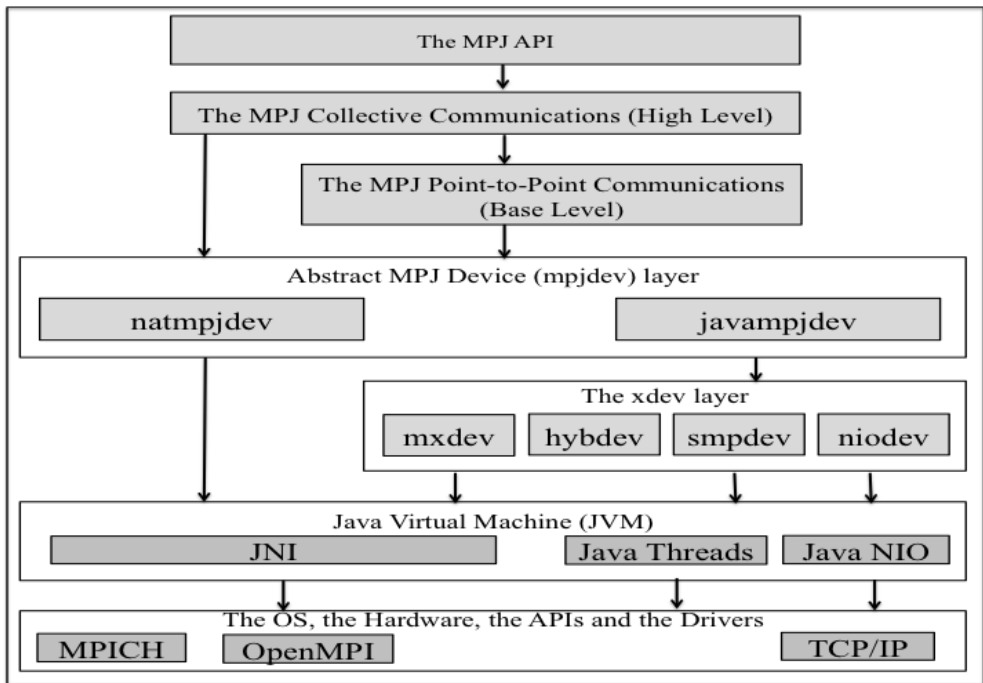



Figura 6: Sistema de camadas do MPJ.

O MPJ Express é capaz de ser executado em dois modos, chamados de modo multicore e modo cluster. No modo cluster, aplicações paralelas são executadas em umtípico ambiente de cluster, em que vários elementos de processamento comunicam entre sí utilizando um meio de interconexão rápido, como Gibabit Ethernet ou outra tecnologia proprietária como Myrinet e Infiniband (QAMAR et al., 2014). É interessante ressaltar que o MPJ permite troca de dados entre redes de tecnologias diferentes. No modo multicore, uma aplicação paralela Java executa em um único sistema composto por memória compartilhada ou processadores multicore.

A arquitetura do MPJ Express apresenta diferentes níveis: a API MPJ, high level, base-level, (*mpjdev*) e *xdev*. As três camadas superiores são acessíveis ao desenvolvedor, para que o mesmo possa implementar aplicações paralelas. A camada base-level contém as primitivas da comunicação ponto-a-ponto como send e recv. As camadas high-level e MPJ API contém as rotinas de comunicação coletiva, tipos de dados derivados e topologias virtuais.

No modelo do MPJ, a camada mais alta e a camada mais baixa dependem das camadas MPJ *device* (*mpjdev*) e *xdev* para estabelecer as comunicações e interações com hardware de rede em questão (LIM et al., 2003). A primeira camada citada provê implementação para o sistema de troca de mensagens que permite a interação com implementações MPI nativas. A segunda camada citada, provê acesso a sockets Java, memória compartilhada, ou bibliotecas de comunicação especializadas.

Ainda analisando a estrutura de camadas do MPJ, já apresentada na Figura 6, a camada *mpjdev* possui duas implementações. A primeira delas é a javampjdev, que provê drivers de comunicação em puro Java, com foco na sua portabilidade. Vale destacar que, como se percebe na Figura 6, que essa implementação faz uso de recursos da camada *xdev*. A segunda implementação é a natmpjdev, que usa JNI wrappers (classes especiais para empacotamento de streams) para a biblioteca nativa MPI.

A camada *xdev* oferece uma interface para desenvolvimento de novos drivers de comunicação para interconexão de redes subjacentes. Atualmente, tal camada contém os seguintes drivers para dispositivos:

- Java New I/O (NIO), conhecido como *niodev*, que é usado para executar programas em clusters usando a tecnologia ethernet.
- dispositivo Myrinet express, conhecido como *mxdev*, que é usado para executar programas em clusters conectados pela tecnologia Myrinet express.
- dispositivo híbrido, conhecido como *hybdev*, que é usado para executar programas em clusters de computadores multicore.
- dispositivo de memória compartilhada, conhecido como *smpdev*, que é usado para executar programas em computadores com memória compartilhada.

GRASP PARALELO

De acordo com (CUNG et al., 2002), a implementação paralela de metaheurísticas é uma alternativa razoável e eficiente para acelerar o processo de busca por boas soluções para problemas difíceis de otimização combinatória. Permitindo não somente resolver problemas maiores ou encontrando soluções de melhor qualidade, mas propiciando métodos mais robustos que sua versão sequencial.

Dado que a metaheurística GRASP é um processo iterativo que consome bem mais tempo que as heurísticas gulosas convencionais (RESENDE; RIBEIRO, 2005), a mesma abre possibilidade para uma implementação paralela efetiva e eficiente.Assim sendo, nesse trabalho, para se obter um melhor resultado que o alcançado por uma única máquina (processamento sequencial), foi utilizado o dispositivo de comunicação niodev em um cluster (COW). A tecnologia de rede ethernet foi utilizada como meio físico para realizar as trocas de mensagens entre os nós do agrupamento.

Fazendo uso de uma implementação Java do MPI (conforme descrito na seção 4.5) foi possível implementar uma versão do algoritmo GRASP proposto para um ambiente paralelo/distribuído.

A próxima seção apresenta uma descrição do software R, utilizado como parâmetro de comparação com as soluções GRASP implementadas (sequencial e paralelo) nesse trabalho. Os pacotes e funções para o cálculo do ccc no R também serão listados. Para finalizar, são apresentados resultados computacionais e análises de modo a verificar os benefícios alcançados com as soluções propostas nesse trabalho.

RESULTADOS E DISCUSSÕES

Após a implementação dos algoritmos propostos por esse trabalho, foi realizado um teste comparativo com um software que permitisse analisar seus desempenhos. Esse software é chamado de competidor, ou seja, aquele cujo resultado será comparado com o das propostas GRASP apresentadas nesse trabalho. Para isso, foram analisados alguns softwares que realizassem o cálculo do *ccc*. Contudo, para se determinar qual seria o escolhido, critérios como ampla utilização, fácil acesso e custo foram utilizados. Nesse sentido, o software competidor escolhido nesse trabalho foi o software R.

O software R é um ambiente de software livre desenvolvido para a manipulação de dados, realização de cálculos estatísticos e gráficos que é muito utilizado na comunidade de estatísticos e mineradores de dados (MUENCHEN, 2012). Neste trabalho, foi utilizado o software R em sua versão 3.1.0.

Tal software possui no pacote **stats** (que contém uma grande quantidade de funções estatísticas) um conjunto de funções que permitem realizar o cálculo do coeficiente de correlação cofenética. Nesse trabalho foram usadas: a função **hclust** para realizar o agrupamento utilizando o método *single linkage*, a função **cophenetic** para gerar a matriz cofenética a partir do agrupamento realizado, e também a função **cor**, que foi utilizada para correlacionar as matrizes fenética e cofenética e assim, calcular o *ccc*.

De modo a exemplificar a diferença dos dendrogramas gerados por ambos os métodos, a Figura 7 mostra os dendrogramas gerados pelo software R e pelo método GRASP proposto para uma matriz de 20 pontos.

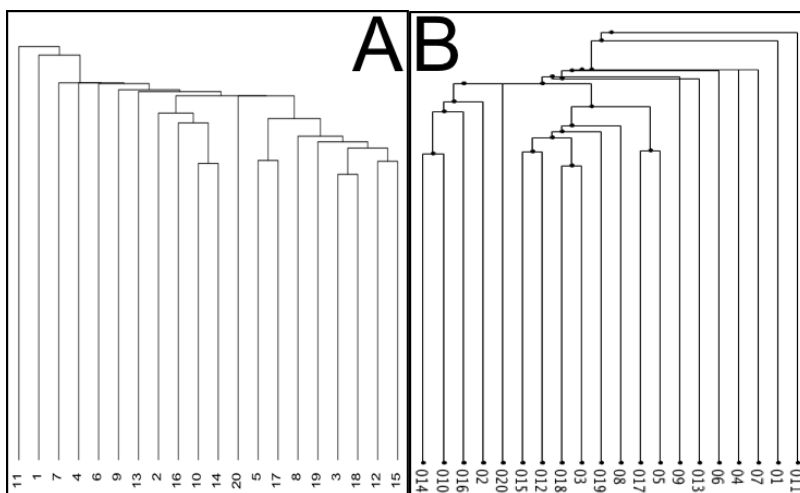



Figura 7: (A) Solução do software R (B) solução GRASP proposta.

O dendrograma da Figura 7(A) foi gerado pelo software R e obteve *ccc* = 0,5026427. Já o dendrograma da Figura 7(B) foi gerado pela solução GRASP sequencial proposta e obteve *ccc* = 0,5030748, configurando-a como melhor que a anterior. Dessa forma, é possível verificar as diferenças gráficas dos dendrogramas e calcular seus coeficientes de correlação cofenética e compará-los.

Contudo, antes de executar os testes, é preciso definir algumas instâncias do algoritmo GRASP. Tal definição também é chamada de calibragem e esse é o assunto abordado na próxima seção.

AS INSTÂNCIAS E O AMBIENTE DE TESTE DA SOLUÇÃO GRASP

O algoritmo foi submetido às instâncias geradas pelo software R. Para esse trabalho foram utilizadas matrizes de 100 (100x100) e 250 (250x250) pontos com valores gerados randomicamente entre 0 e 100, com precisão de 12 casas de decimais, no espaço $\boldsymbol{R}^3$.

O algoritmo proposto foi executado em um computador com a seguinte configuração: processador Intel Core i5 de 2,3 GHz, 8Gb de memória RAM 1333MHz DDR3 e sistema operacional Mac OS X Yosemite, versão 10.10.2. Já o código da heurística GRASP proposta foi implementada em Java e executado na JVM versão 8u40.

Os parâmetros (alfa e número de iterações) do GRASP, foram definidos de forma empírica a partir de testes. Como o mesmo possui um comportamento estocástico, foram realizados uma média dos valores em 20 execuções para cada valor de parâmetro que se deseja testar. Assim, tem-se uma melhor ideia do comportamento do método sobre o conjunto de parâmetros que obtém em média melhores resultados.

Para determinar o melhor valor de alfa (–), a heurística GRASP proposta foi executada com 500 iterações, utilizando como referência uma matriz de 100 pontos e realizando 20 testes para cada valor de alfa considerado. Os valores de alfa considerados para teste foram 0,01 fixo e também variando nos seguintes intervalos: 0,01 - 0,02, 0,01 - 0,03, 0,01 - 0,04, 0,01 - 0,05, 0,01 - 0,09, 0,02 - 0,03, 0,02 - 0,04, 0,02 - 0,05. Desta análise foi verificada que a média entre os 20 testes realizados que obtiveram maior valor coeficiente de correlação cofenética (ccc) em relação ao calculado pelo software R ,para a mesma matriz de distâncias, possuia o alfa fixo em 0,01. Logo em seguida vieram os coeficientes calculados com os alfas entre 0,01 - 0,02, 0,01 - 0,03, 0,01 - 0,04, 0,01 - 0,05, 0,02 - 0,04, 0,01 - 0,09, 0,02 - 0,03 e 0,02 - 0,05. Com base nesses testes, a valor de alfa utilizado foi fixado em 0,01.

Para determinar a melhor quantidade de iterações para as matrizes de 100 pontos, foram realizados 20 testes para cada quantidade. Os valores das iterações testados foram

200, 500, 700, 1000, 1200, 1500, 1600, 1700 e 2000. Nessas verificações, percebeu-se que a melhor média dos *ccc*'s foi alcançada com os testes que executavam 1600 iterações. Contudo, calculando a média e considerando o número da iteração em que foi obtido o melhor *ccc*, chegou-se ao valor de 724,8 interações. Dessa forma, para realização dos testes, considerando uma boa relação entre o esforço computacional e a qualidade das soluções obtidas, determinou-se 800 para o valor das iterações. Assim sendo, todos os testes realizados a seguir com o algoritmo GRASP proposto considera um alfa fixo de 0,01 e 800 iterações do loop principal. O desempenho medido foi calculado através da média aritmética de 20 execuções sobre cada instância de teste.

RESULTADOS OBTIDOS PELA SOLUÇÃO GRASP SEQUENCIAL

Nesta seção, serão apresentados os resultados computacionais obtidos pela implementação do GRASP proposto para o problema em questão, fazendo uso das instâncias e especificações conforme descrito na seção 5.1.

Tanto na Tabela 10 quanto na Tabela 11, a primeira coluna apresenta o coeficiente de correlação cofenética calculada pelo Software R. A segunda coluna apresenta a média dos *ccc*'s calculados pelo algoritmo GRASP. A terceira coluna apresenta o tempo médio, em segundos, de execução do GRASP. Já a última coluna apresenta, em porcentagem, o valor médio da diferença obtida pelo algoritmo GRASP sequencial proposto em relação ao método *single linkage* executado pelo software R.

Na Tabela 10 é feita uma comparação entre o coeficiente de correlação cofenética calculado pelo software R e o coeficiente calculado pelo algoritmo GRASP sequencial, para um conjunto de 10 instâncias de testes com 100 pontos gerados aleatoriamente.

Coeficiente R	Coeficiente GRASP Sequencial	Tempo Médio	GAP
0,5550619	0,5550619	738,00	0,000%
0,4997008	0,5016502	613,80	0,390%
0,4857282	0,4884351	628,20	0,557%
0,5075269	0,5078410	633,60	0,062%
0,5369218	0,5443324	687,00	1,380%
0,5810349	0,5819616	664,20	0,159%
0,5589223	0,5614827	708,60	0,458%
0,5047959	0,5116431	606,00	1,356%
0,5256531	0,5311426	654,60	1,044%
0,5644307	0,5644620	751,80	0,006%

Tabela 10: Desempenho do algoritmo GRASP para matrizes de 100 pontos.

Observando-se os resultados da Tabela 10, é possível verificar que o valor médio

dos coeficientes de correlação cofenética obtidos pelo GRASP Sequencial são, em 90% dos testes, melhores que os obtidos pelo Software R utilizando o método *single linkage*. Em 10% dos testes o valor do coeficiente é igual ao calculado pelo software R. Também se pode afirmar que o coeficiente obtido é, no mínimo, igual ao do programa utilizado como referência.

E na Tabela 11 é feita uma comparação entre o coeficiente de correlação cofenética calculado pelo software R e o coeficiente calculado pelo algoritmo GRASP Sequencial, para um conjunto de 10 instâncias de testes com 250 pontos gerados aleatoriamente.

Coeficiente R	Coeficiente GRASP Sequencial	Tempo Médio	GAP
0,5647190	0,5647190	72828,00	0,000%
0,5347004	0,5353080	72039,60	0,114%
0,5023091	0,5023422	73335,60	0,007%
0,5041196	0,5101581	71539,20	1,198%
0,5139481	0,5163616	72040,32	0,470%
0,5070545	0,5093753	71319,60	0,458%
0,5201755	0,5215701	74818,80	0,268%
0,5068115	0,5076109	73357,20	0,158%
0,4871727	0,4890002	72356,40	0,375%
0,5392797	0,5392797	68752,80	0,0%

Tabela 11: Desempenho do algoritmo GRASP para matrizes de 250 pontos.

Analisando os resultados da Tabela 11, se percebe que o ccc obtido pelo GRASP Sequencial foi, em 80% dos casos, melhor que o obtido pelo R. Em 20% dos outros casos, o coeficiente obtido foi igual ao calculado pelo R. Em nenhum deles o coeficiente obtido foi inferior ao calculado pelo programa de referência. Como pode ser verificado através dos resultados, em muitos casos foi possível melhorar o coeficiente em questão, utilizando a solução GRASP sequencial.

Das Tabelas 10 e 11 é possível observar que o GRASP proposto obteve soluções no mínimo tão boas quanto o método *Single Linkage*, sendo melhor na maioria das vezes. Isso com baixo tempo computacional. Uma análise mais detalhada dos resultados de tais tabelas será apresentada na seção 5.5.

Da mesma forma que foi preciso configurar e calibrar a solução GRASP Sequencial, o mesmo deve ser feito para a solução GRASP Paralela. Esse é o assunto abordado na próxima seção.

AS INSTÂNCIAS E O AMBIENTE DE TESTE DA SOLUÇÃO GRASP PARALELA

Com o objetivo de obter um melhor desempenho do algoritmo GRASP proposto,

uma implementação paralela do mesmo foi desenvolvida. As mesmas instâncias geradas e utilizadas nos testes anteriores, ou seja, as matrizes de 100 e 250 pontos (conforme especificado na seção 5.1), foram utilizadas também no caso paralelo para comparação.

O GRASP paralelo implementado foi executado em um cluster de computadores (COW) formado por 11 máquinas, todas com a seguinte configuração: processador AMD Phenom (tm) II X2 550 com dois núcleos, 4Gb de memória RAM 1333MHz DDR3 e sistema operacional Ubuntu 14.04 LTS A . A Figura 8 apresenta uma representação do ambiente utilizado nos testes. O código da solução proposta foi implementado em Java e executado na JVM versão 25.45-b02.

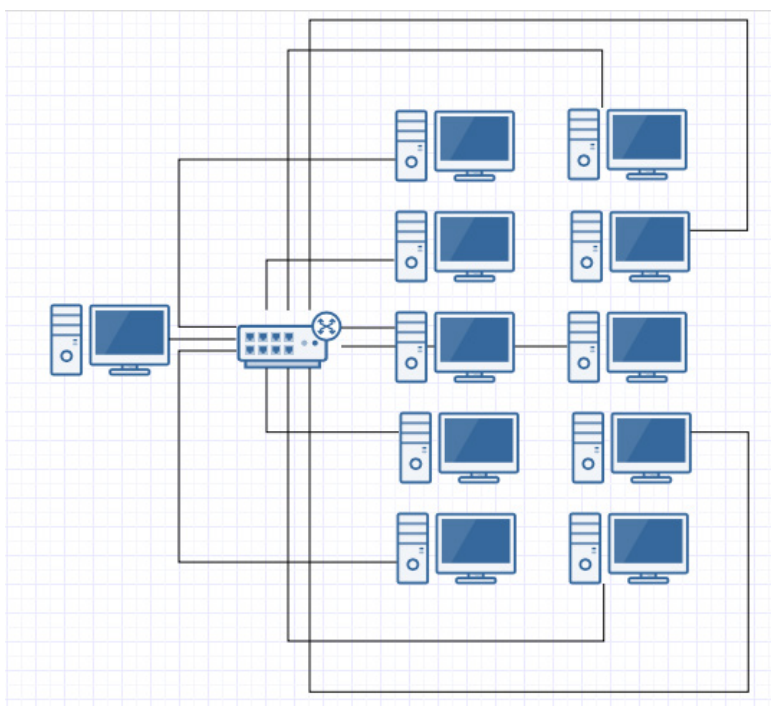



Figura 8: Cluster utilizado para executar a solução GRASP Paralela.

Na Figura 8, o computador mais a esquerda, vamos chamá-lo de mestre, é responsável por iniciar a execução do programa e também por realizar parte do processamento. A quantidade total de iterações são divididas pelas 11 máquinas do cluster. Ao final do processamento, cada máquina retorna o melhor *ccc* que obteve para o mestre, que compara todos os 11 valores e considera apenas o melhor.

De modo a executar o GRASP paralelo implementado, é necessário instalar e configurar o ambiente paralelo e distribuído. Para esse fim, foi necessário baixar o pacote MPJ Express, na sua versão 0_43, disponível no seguinte endereço http://sourceforge.

net/projects/mpjexpress/files/releases/. Depois, o arquivo com a extensão ".jar" (que é um arquivo java compactado, contendo as classes compiladas e metadados) do programa referente ao GRASP paralelo foi copiado para todas as máquinas do cluster. Também foi copiado para cada máquina, os arquivos a serem utilizados como entrada do programa. Além disso, foi criado um arquivo texto chamado "machines", contendo os endereços IPs de todas as máquinas do cluster.

Com o ambiente de execução paralelo configurado, a seguinte sequencia de comandos, dependendo da finalidade, pode ser executada no terminal do Ubuntu:

- **mpjboot machines** - utilizado para iniciar os daemons do MPJ, nas máquinas com IPs especificados no arquivo machines. Esses daemons proporcionam o middleware necessário à execução paralela no ambiente distribuído no cluster.
- **mpjrun.sh -np 11 -dev niodev grasp** - utilizado para executar o programa no cluster. O parâmetro -np é utilizado para especificar a quantidade de máquinas de processos que serão utilizados(no geral, um por máquina). O parâmetro **-dev** é utilizado para especificar qual driver de interconexão será utilizado pelo programa. Já **grasp** é o nome do arquivo ".jar" referente ao método que será executado.
- **mpjhalt machines** - no final da execução dos testes, utilizou-se esse comando para parar os daemons nas máquinas especificadas no arquivo machines e finalizar o ambiente paralelo.

As mesmas matrizes de distâncias utilizadas como entrada para o algoritmo GRASP Sequencial, foram também submetidas ao algoritmo GRASP Paralelo. Os resultados obtidos nesses testes são apresentados na próxima seção.

RESULTADOS OBTIDOS PELA SOLUÇÃO GRASP PARALELA

As tabelas 12 e 13 apresentam o resultado dos testes executados nessa solução paralela. Os campos contidos nessa tabelas são os mesmos descritos na seção 5.2 para as tabelas 10 e 11.

Na Tabela 12 é feita uma comparação entre o coeficiente de correlação cofenético calculado pelo software R e o coeficiente calculado pelo algoritmo GRASP Paralelo, para um conjunto de 10 instâncias de testes com 100 pontos gerados aleatoriamente.

Coeficiente R	Coeficiente GRASP Paralelo	Tempo Médio	GAP
0,5550619	0,5550619	184,51	0,000%
0,4997008	0,5035449	153,45	0,769%
0,4857282	0,4906651	158,00	1,016%
0,5075269	0,5089114	158,50	0,273%
0,5369218	0,5460952	171,55	1,709%
0,5810349	0,5840950	166,15	0,527%
0,5589223	0,5659074	177,05	1,250%
0,5047959	0,5334992	151,75	5,686%
0,5256531	0,5334992	163,36	1,493%
0,5644307	0,5645401	187,94	0,019%

Tabela 12: Desempenho do algoritmo GRASP Paralelo para matrizes de 100 pontos.

Analisando os resultados da Tabela 12, se percebe que o ccc obtido pelo GRASP Paralelo foi, em 90% dos casos, melhor que o obtido pelo R. Em 10% dos outros casos, o coeficiente obtido foi igual ao calculado pelo R. Em nenhum deles o coeficiente obtido foi inferior ao calculado pelo programa de referência. Assim, se pode afirmar que o GRASP Paralelo consegue obter ccc melhores que os do método *single linkage*.

Na Tabela 13 é feita uma comparação entre o coeficiente de correlação cofenético calculado pelo software R e o coeficiente calculado pelo algoritmo GRASP Paralelo, para um conjunto de 10 instâncias de testes com 250 pontos gerados aleatoriamente.

Coeficiente R	Coeficiente GRASP Paralelo	Tempo Médio	GAP
0,5647190	0,5647190	14565,60	0,000%
0,5347004	0,5367396	14407,92	0,381%
0,5023091	0,5023382	14667,12	0,006%
0,5041196	0,5116976	14307,44	1,503%
0,5139481	0,5175755	14408,46	0,706%
0,5070545	0,5107083	14263,91	0,721%
0,5201755	0,5227327	14962,76	0,492%
0,5068115	0,5083424	14671,04	0,302%
0,4871727	0,4902774	14471,58	0,637%
0,5392797	0,5395972	13750,46	0,059%

Tabela 13: Desempenho do algoritmo GRASP Paralelo para matrizes de 250 pontos.

Ao verificar os resultados da Tabela 13, se percebe que o ccc obtido pelo GRASP Paralelo foi, em 90% dos casos, melhor que o obtido pelo R. Em 10% dos outros casos, o coeficiente obtido foi igual ao calculado pelo R. Em nenhum deles o coeficiente obtido foi inferior ao calculado pelo programa de referência. Novamente, a solução GRASP Paralelo

consegue obtém melhores *ccc*'s que os do método *single linkage*.

Após a realização dos testes, é possível afirmar que em nenhum deles o coeficiente obtido foi inferior ao calculado pelo programa de referência. E em muitos dos casos foi possível melhorar tal coeficiente, tanto com a solução GRASP sequencial quanto com a solução GRASP paralela. Tais resultados mostram que a implementação GRASP realizada nesse trabalho é bastante promissora quando aplicada ao problema de clusterização hierárquica.

Além dessas matrizes de 100 e 250 pontos, para verificar o desempenho do algoritmo proposto, foram utilizadas três matrizes como entrada de dados, uma de 500 (500x500) pontos, outra com 1000 (1000x1000)pontos e outra de 10000(10000x10000), que foram criadas da mesma forma que as de 100 e 250 pontos, conforme descrito na seção 5.1. Essas matrizes de dimensões maiores foram criadas de modo a verificar a robustez e capacidade do método paralelo proposto de resolver problemas grandes em tempo computacional razoável.

Na Tabela 14 é feita uma comparação entre o coeficiente de correlação cofenético calculado pelo software R e o coeficiente calculado pelo algoritmo GRASP Paralelo, para um conjunto de 10 instâncias de testes com 500 e 1000 pontos, e 1 instância de 10000 pontos gerados aleatoriamente.

Quantidade de pontos	Coeficiente R	GRASP Paralelo	Tempo Médio	GAP
500	0,5349349	0,5415145	24303,60	0,123%
1000	0,5338410	0,5392328	42531,30	0,101%
10000	-	0.5188314	92581,38	-

Tabela 14: Desempenho do algoritmo GRASP Paralelo para matrizes de 500, 1000 e 10000 pontos.

O coeficiente de correlação cofenética obtido nas matrizes de 500 e 1000 pontos, na média, foi melhor que o gerado pelo software R. Considerando uma única execução, o valor do *ccc* foi sempre, no mínimo igual ao calculado pelo software R. No teste realizado com a matriz de 10000 pontos como entrada, o software R não foi capaz de calcular o *ccc*. O mesmo finaliza sua execução, e não consegue completá-la, devendo-se a extrapolação do limite de memória das estruturas de dados e estouro de pilha na execução do método *single linkage*. Contudo, a solução paralela obteve êxito no cálculo do coeficiente para essa matriz. Esses testes serviram para verificar a eficácia da solução GRASP Paralela em trabalhar com matrizes de distâncias de grandes dimensões como entrada.

Outra análise feita com os resultados obtidos, foi comparar os coeficientes obtidos para as mesmas matrizes pelos algoritmos propostos. Na Tabela 15 é feita uma comparação

entre o coeficiente de correlação cofenético calculado pelo GRASP Sequencial e o coeficiente calculado pelo algoritmo GRASP Paralelo, para um conjunto de 10 instâncias de testes 100 pontos gerados aleatoriamente.

GRASP Sequencial	GRASP Paralelo	GAP
0,5550619	0,5550619	0,000%
0,5016502	0,5035449	0,380%
0,4884351	0,4906651	0,460%
0,5078410	0,5089114	0,210%
0,5443324	0,5460952	0,320%
0,5819616	0,5840950	0,370%
0,5614827	0,5659074	0,790%
0,5116431	0,5334992	4,270%
0,5311426	0,5334992	0,440%
0,5644620	0,5645401	0,010%

Tabela 15: Comparativo dos coeficientes obtidos para matrizes de 100 pontos.

Analisando os resultados da Tabela 15, se percebe que o ccc obtido pelo GRASP Paralelo foi, em 90% dos casos, melhor que o obtido pelo GRASP Sequencial. Em 10% dos outros casos, os coeficientes obtidos foram iguais. Assim, se pode afirmar que o GRASP Paralelo consegue obter *ccc* melhores que GRASP Sequencial.

Na Tabela 16 é feita uma comparação entre o coeficiente de correlação cofenético calculado pelo GRASP Sequencial e o coeficiente calculado pelo algoritmo GRASP Paralelo, para um conjunto de 10 instâncias de testes 250 pontos gerados aleatoriamente.

GRASP Sequencial	GRASP Paralelo	GAP
0,5647190	0,5647190	0,000%
0,5353080	0,5367396	0,270%
0,5023422	0,5023382	0,000%
0,5101581	0,5116976	0,300%
0,5163616	0,5175755	0,240%
0,5093753	0,5107083	0,260%
0,5215701	0,5227327	0,220%
0,5076109	0,5083424	0,140%
0,4890002	0,4902774	0,260%
0,5392797	0,5395972	0,060%

Tabela 16: Comparativo dos coeficientes obtidos para matrizes de 250 pontos.

Analisando os resultados da Tabela 15, se percebe que o *ccc* obtido pelo GRASP Paralelo foi, em 80% dos casos, melhor que o obtido pelo GRASP Sequencial. Em 20% dos

outros casos, os coeficientes obtidos foram iguais. Assim, se pode afirmar que o GRASP Paralelo consegue obter ccc melhores que GRASP Sequencial.

Como pode ser verificado nos comparativos realizados nas tabelas 15 e 16, a solução GRASP Paralela se mostrou ainda mais eficiente que a solução sequencial, não só no que diz respeito ao tempo computacional, mas também proporcionou um *ccc* ainda melhor na maioria dos casos. Essa melhora no *ccc* é devido ao gerador de números randômicos utilizados na implementação do GRASP para escolher um elemento na LRC. No GRASP Sequencial há um gerador para cada execução do programa. Já no GRASP Paralelo, há 11 geradores randômicos para uma execução do código, um para cada nó do cluster de computadores. Essa diferença apresentada na solução paralela permite explorar melhor a característica randômica da metaheurística e melhorar o *ccc* dos agrupamentos gerados.

Após analisar o coeficientes obtidos em relação ao software R, foi feita uma comparação entre o tempo médio de execução dos testes da solução GRASP sequencial com a solução GRASP paralela. A tabela 17 apresenta um comparativo entre os tempos obtidos nos testes com as matrizes de 100 pontos.

GRASP Sequencial	GRASP Paralelo	GAP
738,00	184,51	-74,999%
613,80	153,45	-75,000%
628,20	158,00	-74,849%
633,20	158,50	-74,968%
687,00	171,55	-75,029%
664,20	166,15	-74,985%
708,60	177,05	-75,014%
606,00	151,75	-74,959%
654,60	163,36	-75,044%
751,80	187,94	-75,001%

Tabela 17: Comparação dos tempos de execução para matrizes de 100 pontos.

Como pode ser visto na Tabela 17 para as matrizes de 100 pontos, a solução paralela conseguiu concluir seu processamento em um tempo muito inferior ao da solução executada em uma única máquina. O tempo foi reduzido em pelos menos 74% o seu valor.

E a tabela 18 apresenta um comparativo entre os tempos obtidos nos testes com as matrizes de 250 pontos.

GRASP Sequencial	GRASP Paralelo	GAP
72828,00	14565,60	-80,000%
72039,60	14407,92	-80,000%
73335,60	14667,12	-80,000%
71539,20	14307,44	-80,001%
72040,32	14408,46	-79,999%
71319,60	14263,91	-80,000%
74818,80	14962,76	-80,001%
73357,20	14671,04	-80,001%
72356,40	14471,28	-80,000%
68752,80	13750,46	-80,000%

Tabela 18: Comparação dos tempos de execução para matrizes de 250 pontos.

Como pode ser visto na Tabela 18 para as matrizes de 250 pontos, também foi apresentado um desempenho muito superior ao da outra proposta GRASP. Para essas matrizes, o tempo foi reduzido em pelo menos 79% em seu valor. Analisando esses resultados, é correto afirmar que a solução paralela, para matrizes de 100 e 250 pontos, conseguiu na maioria dos testes melhorar o *ccc* em um tempo bem reduzido, se tornando assim a melhor das soluções.

Nos resultados apresentados nesses testes, pode-se verificar que a metaheurística GRASP é muito promissora quando aplicada ao problema de clusterização hierárquica. Os testes obtiveram melhores resultados (maiores valores para os *ccc*'s) na maioria das vezes. Nunca o resultado foi pior que o obtido pelo método *single linkage* calculado pelo software competidor. Utilizando puramente o coeficiente de correlação cofenética como parâmetro, é verdadeira a afirmação que o algoritmo GRASP implementado nesse trabalho, e utilizando o método *single linkage* como heurística subordinada é melhor do que o método *single linkage* puramente guloso.

Entre os algoritmos desenvolvidos nesse trabalho, a solução paralela apresentou melhores resultados que os obtidos pela solução sequencial. E isso foi obtido em um tempo muito menor. O algoritmo que fez uso do processamento paralelo se mostrou superior na grande maioria dos testes, tanto no quesito melhor *ccc* quanto no quesito menor tempo. Isso torna essa implementação a melhor.

Como pode ser visto pelos resultados aqui apresentados, os objetivos traçados para esse trabalho foram alcançados. A aplicação de metaheurísticas na clusterização hierárquica obteve um desempenho muito bom, o que nos motiva a explorar ainda mais esse assunto.

Na próxima seção são feitas as considerações finais sobre esse trabalho e também metas para trabalhos futuros são delineadas.

CONCLUSÕES E TRABALHOS FUTUROS

Nesse trabalho foi estudado o problema de clusterização hierárquica utilizando o método *single linkage*. Como contribuição, foi desenvolvido um algoritmo utilizando a metaheurística GRASP para tal problema, visando gerar soluções que obtivessem melhor *ccc* que o método single linkage convencional.

Posteriormente, objetivando-se trabalhar com matrizes maiores, foi implementada uma solução paralela utilizando a mesma metaheurística. Tal solução fez uso de uma rede composta por 11 computadores em COW para a realização dos testes computacionais. Os dendrogramas gerados pelos diferentes métodos foram analisados através da averiguação do seu coeficiente de correlação cofenética.

Todos os métodos (*single linkage*, GRASP sequencial e GRASP paralelo) foram testados inicialmente em 20 matrizes de distâncias (10 com 100 e outras 10 com 250 pontos no R3), onde os métodos GRASP aqui propostos, tiveram destaque em termos da qualidade das soluções geradas, obtendo soluções no mínimo tão boas quanto as proporcionadas pelo método *single linkage* e sendo, em sua maioria, ainda melhor. Entre as duas soluções implementadas, considerando apenas o coeficiente de correlação cofenética, a melhor foi a GRASP Paralela, pois alcançou melhores *ccc* que a solução sequencial.

A solução GRASP Paralelo também foi submetida a outros testes com matrizes maiores (500, 1000 pontos no R3), onde também foi verificado seu êxito (sempre obtendo soluções de melhor qualidade). Quando foi utilizada uma matriz de 10000 como entrada, o GRASP Paralelo conseguiu concluir seu processamento e retornar o melhor *ccc* obtido, o que software R não foi capaz de fazer, finalizando sua execução antes de completar seu cálculo.

Um outro aspecto relevante foi o tempo computacional satisfatório, principalmente os obtidos pela solução paralela quando comparada com a solução sequencial. A redução do tempo de processamento foi de pelo menos 74% para as matrizes de 100 pontos, e 79% para as matrizes de 250 pontos. Isso foi possível porque foi utilizado como ambiente paralelo, um cluster formado por 11 computadores, que dividiam de forma igual a carga de processamento imposta pela solução GRASP. Se uma cluster com mais computadores fosse utilizado, os resultados poderiam ser ainda melhores.

As recomendações de continuidade para este trabalho são no sentido de tornar o algoritmo mais robusto e eficiente. Posteriormente, objetivando trabalhar com matrizes maiores, recomenda-se:

- Realização de testes e comparações fazendo uso de bases de dados reais.

- Utilização da técnica de *path-relinking* juntamente com a metaheurística GRASP.

Embora as estratégias utilizadas para o problema da clusterização hierárquica tenham sido limitadas ao método GRASP puro, acredita-se que soluções melhores a estas poderão ser obtidas para outras variantes do método. Portanto, outra recomendação seria o estudo de variantes do método GRASP, tais como as propostas por (FLEURENT; GLOVER, 1999) e (ROSSETI, 2003).

REFERÊNCIAS

ALDENDERFER, M. S. Methods of cluster validation for archaeology. *World Archaeology*, Taylor & Francis, v. 14, n. 1, p. 61–72, 1982.

ALVARENGA, F.; ROCHA, M. L. Melhorando o desempenho da metaheurística grasp utilizando a técnica path-relinking: Uma aplicação para o problema da árvore geradora de custo mínimo com grupamentos. *XXXVIII Simpósio Brasileiro de Pesquisa Operacional*, 2006.

ANDERBERG, M. R. *Cluster Analysis for Applications: Probability and Mathematical Statistics: A Series of Monographs and Textbooks*. [S.l.]: Academic press, 2014. v. 19.

ANDREOPOULOS, B. et al. A roadmap of clustering algorithms: finding a match for a biomedical application. *Briefings in Bioinformatics*, Oxford Univ Press, v. 10, n. 3, p. 297–314, 2009.

ARABIE, P.; HUBERT, L. J. An overview of combinatorial data. *Clustering and classification*, World Scientific, p. 5, 1996.

BAKER, M.; CARPENTER, B.; SHAFT, A. Mpj express: towards thread safe java hpc. In: IEEE. Cluster Computing, *2006 IEEE International Conference on*. [S.l.], 2006. p. 1–10.

BANDYOPADHYAY, S.; COYLE, E. J. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In: IEEE. INFOCOM 2003. *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*. [S.l.], 2003. v. 3, p. 1713–1723.

BARKER, B. Message passing interface (mpi). In: *Workshop: High Performance Computing on Stampede*. [S.l.: s.n.], 2015.

BEGUELIN, A. et al. *A users' guide to PVM (parallel virtual machine)*. [S.l.], 1991.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, ACM, v. 35, n. 3, p. 268–308, 2003.

BOUGUETTAYA, A. et al. Efficient agglomerative hierarchical clustering. *Expert Systems with Applications*, Elsevier, v. 42, n. 5, p. 2785–2797, 2015.

CASTRO, A. A. M. D.; PRADO, P. P. L. D. Algoritmos para reconhecimento de padrões. *Revista Ciências Exatas*, v. 8, n. 2002, 2001.

CUNG, V.-D. et al. Strategies for the parallel implementation of metaheuristics. In: *Essays and surveys in metaheuristics*. [S.l.]: Springer, 2002. p. 263–308.

DASH, M. et al. Fast hierarchical clustering and its validation. *Data & Knowledge Engineering*, Elsevier, v. 44, n. 1, p. 109–138, 2003.

DENG, Y.; KOROBKA, A. The performance of a supercomputer built with commodity components. *Parallel Computing*, Elsevier, v. 27, n. 1, p. 91–108, 2001.

DIAS, C. R.; OCHI, L. S. Desenvolvimento e analise experimental de algoritmos evolutivos para o problema de clusterizacao automatica em grafos orientados. In: In: *I Brazilian Workshop on Evolutionary Computation*. [S.l.: s.n.], 2004. v. 1, p. 11–18.

EL-REWINI, H.; ABD-EL-BARR, M. Message passing interface (mpi). *Advanced computer architecture and parallel processing*, Wiley Online Library, p. 205–233, 2005.

EVERITT, B. S. Unresolved problems in cluster analysis. *Biometrics*, JSTOR, p. 169–181, 1979.

FARRIS, J. S. On the cophenetic correlation coefficient. *Systematic Biology*, Oxford University Press, v. 18, n. 3, p. 279–285, 1969.

FEO, T. A.; RESENDE, M. G. Greedy randomized adaptive search procedures. *Journal of global optimization*, Springer, v. 6, n. 2, p. 109–133, 1995.

FESTA, P.; RESENDE, M. G. Effective application of grasp. *Wiley encyclopedia of operations research and management science*, Wiley Online Library, 2009.

FILHO, A. C.; RIBEIRO, N. D.; BURIN, C. Consistência do padrão de agrupamento de cultivares de feijão conforme medidas de dissimilaridade e métodos de agrupamento. *Pesquisa Agropecuária Brasileira*, SciELO Brasil, v. 45, n. 3, p. 236–243, 2010.

FLEURENT, C.; GLOVER, F. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, INFORMS, v. 11, n. 2, p. 198–204, 1999.

GEIST, A. PVM: *Parallel virtual machine: a users' guide and tutorial for networked parallel computing*. [S.l.]: MIT press, 1994.

GROPP, W.; LUSK, E.; SKJELLUM, A. *Using MPI: portable parallel programming with the message-passing interface*. [S.l.]: MIT press, 1999. v. 1.

HAN, J.; KAMBER, M.; PEI, J. *Data mining: concepts and techniques: concepts and techniques*. [S.l.]: Elsevier, 2011.

HARTIGAN, J. A. *Clustering algorithms (probability & mathematical statistics)*. [S.l.]: John Wiley & Sons Inc, 1975.

HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. [S.l.]: MIT press, 1992.

HOUTE, B. P. V.; HERINGA, J. Accurate confidence aware clustering of array cgh tumor profiles. *Bioinformatics*, Oxford Univ Press, v. 26, n. 1, p. 6–14, 2010.

HRUSCHKA, E. R. *Algoritmos genéticos de agrupamento para extração de regras de redes neurais*. Tese (Doutorado) — UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, 2001.

IGNACIO, A.; FILHO, V. F.; GALVÃO, R. Métodos heurísticos num entorno paralelo. *Simpósio Brasileiro de Pesquisa Operacional*, v. 32, p. 769–788, 2000.

IGNÁCIO, A. A. V.; FILHO, V. J. M. F. Mpi: uma ferramenta para implementação paralela. *Pesquisa Operacional*, SciELO Brasil, v. 22, n. 1, p. 105–116, 2002.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. *ACM computing surveys (CSUR)*, Acm, v. 31, n. 3, p. 264–323, 1999.

JOHNSON, R. A.; WICHERN, D. W. et al. *Applied multivariate statistical analysis*. [S.l.]: Prentice hall Englewood Cliffs, NJ, 1992. v. 4.

JOHNSON, S. C. Hierarchical clustering schemes. *Psychometrika*, Springer, v. 32, n. 3, p. 241–254, 1967.

LASTRA, I. et al. The classification of first episode schizophrenia: a cluster-analytical approach. *Acta Psychiatrica Scandinavica*, Wiley Online Library, v. 102, n. 1, p. 26–31, 2000.

LIM, S. B. et al. A device level communication library for the hpjava programming language. In: *Fifteenth IASTED International Conference on Parallel and Distributed Computing and Systems*. [S.l.: s.n.], 2003. v. 1, p. 314–319.

MAKRETSOV, N. A. et al. Hierarchical clustering analysis of tissue microarray immunostaining data identifies prognostically significant groups of breast carcinoma. *Clinical cancer research*, AACR, v. 10, n. 18, p. 6143–6151, 2004.

METZ, J. *Interpretação de clusters gerados por algoritmos de clustering hierárquico*. Tese (Doutorado) — Universidade de São Paulo, 2006.

MORELLI, C.; VIEIRA, L. Análise de métodos heurísticos de característica gulosa. *IV Semana Acadêmica do PPGC (Programa de Pós-Graduação em Computação)*, 1999.

MUENCHEN, R. A. The popularity of data analysis software. *r4stats. com, http://r4stats. com/articles/popularity/, last accessed*, v. 28, 2012.

NG, R. T.; HAN, J. Clarans: A method for clustering objects for spatial data mining. *Knowledge and Data Engineering, IEEE Transactions on,* IEEE, v. 14, n. 5, p. 1003–1016, 2002.

OCHI, L. S.; DIAS, C. R.; SOARES, S. S. F. Clusterização em mineração de dados. *Instituto de Computação-Universidade Federal Fluminense-Niterói*, 2004.

QAMAR, B. et al. Design and implementation of hybrid and native communication devices for java {HPC}. *Procedia Computer Science*, v. 29, p. 184 – 197, 2014. ISSN 1877-0509. 2014 International Conference on Computational Science. Disponível em: <http://www.sciencedirect.com/science/article/pii/S187705091400194X>.

RESENDE, M. G.; RIBEIRO, C. C. 14 parallel greedy randomized adaptive search procedures. *Parallel Metaheuristics: A new class of algorithms*, John Wiley & Sons, v. 47, p. 315, 2005.

RESENDE, M. G.; RIBEIRO, C. C. Grasp: greedy randomized adaptive search procedures. In: *Search methodologies*. [S.l.]: Springer, 2014. p. 287–312.

ROSSETI, I. *Estratégias sequenciais e paralelas de GRASP com reconexão por caminhos para o problema de síntese de redes a 2-caminhos*. Tese (Doutorado) — Tese de doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2003.

SHAFI, A.; MANZOOR, J. Towards efficient shared memory communications in mpj express. In: IEEE. *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. [S.l.], 2009. p. 1–7.

SOKAL, R. R.; ROHLF, F. J. The comparison of dendrograms by objective methods. *Taxon*, JSTOR, p. 33–40, 1962.

SRINIVAS, M.; MOHAN, C. K. Efficient clustering approach using incremental and hierarchical clustering methods. In: IEEE. *Neural Networks (IJCNN), The 2010 International Joint Conference on*. [S.l.], 2010. p. 1–7.

TANENBAUM, A. S. *Structured computer organization*. [S.l.]: Pearson, 2006.

TOMAR, A. et al. Parallel implementation of machine translation using mpj express. In: IEEE. *Parallel Computing Technologies (PARCOMPTECH), 2013 National Conference on*. [S.l.], 2013. p. 1–5.

TORRES, R. A. *Aplicação de Métodos de Clusterização em um estudo sobre o Mercado Acionário Brasileiro*. Tese (Doutorado) — PUC-Rio, 2013.

TRINDADE, A. Metaheuristicas para o problema de clusterização de células de manufatura. orientador: Luiz satoru ochi. programa de pós grad. *Computação, IC/UFF*, 2004.

TRINDADE, A.; OCHI, L. An efficient evolutionary algorithm for the manufacturing cell design problem. *Proc.(CD-ROM) of the XII CLAIO, Havana, Cuba*, 2004.

WEBB, A. R. *Statistical pattern recognition*. [S.l.]: John Wiley & Sons, 2003.

WILKINSON, B.; ALLEN, M. *Parallel programming*. [S.l.]: Prentice hall New Jersey, 1999. v. 999.

APÊNDICE A – PÔSTER PUBLICADO: XLVII SBPO

Metaheurística GRASP Aplicada ao Problema de Clusterização Hierárquica pelo Método Single Linkage

Napoleão Póvoa Ribeiro Filho
Mestrado em Modelagem Computacional de Sistemas - Universidade Federal do Tocantins
Av. NS 15, S/N – ALCNO 14 - Bloco Bala 2 – Sala 01, CEP: 77020-210, Palmas - TO

Marcelo Lisboa Rocha
Mestrado em Modelagem Computacional de Sistemas - Universidade Federal do Tocantins
Av. NS 15, S/N – ALCNO 14 - Bloco Bala 2 – Sala 01, CEP: 77020-210, Palmas - TO

RESUMO

Este trabalho traz uma nova proposta para melhora da clusterização hierárquica pelo método *single linkage*, utilizando como parâmetro o coeficiente de correlação cofenética (ccc). Na clusterização hierárquica, parte-se de uma matriz de dissimilaridades e utilizando como critério a menor distância entre os pontos, onde os mesmos vão sendo agrupados até gerar ao final uma árvore invertida conhecida como dendrograma. O ccc, obtido após a construção do dendrograma, indica o quão fiel está o agrupamento em relação a matriz de dados originais. Com o objetivo de gerar dendrogramas que resultem em melhores ccc, propõe-se no presente trabalho um novo algoritmo que utiliza os conceitos da metaheurística GRASP. Testes foram realizados para comprovar o desempenho do algoritmo proposto, comparando os coeficientes obtidos com os gerados pelo software R.

Palavras Chave: GRASP, Clusterização Hierárquica, Coeficiente de Correlação Cofenética

ABSTRACT

This paper presents a new proposal for improved hierarchical clustering by single linkage method, using as parameter cophenetic correlation coefficient (ccc). In hierarchical clustering, it's used a dissimilarity matrix and using as a criterion the shortest distance between points, which are grouped and at the end generate an inverted tree known as dendrogram. The ccc, after the construction of the dendrogram indicates how faithful is the grouping as regards the original data matrix. In order to generate dendrograms that result in better ccc, is propose in this paper a new algorithm that uses the concepts of GRASP. Tests were conducted to confirm the performance of the proposed algorithm by comparing the coefficients obtained with those generated by software R.

Keywords: GRASP, Hierarchical clustering, Cophenetic Correlation Coefficient

SOBRE O AUTOR

NAPOLEÃO PÓVOA RIBEIRO FILHO - Nasceu em Dianópolis, Tocantins, em 1980. Possui formação em Engenharia de computação, pós-graduação em Banco de Dados e mestrado em Modelagem Computacional de Sistemas. Atuou como desenvolvedor na linguagem de programação NATURAL. Tem mais de 16 anos de experiência como docente em algumas instituições de ensino, ministrando as cadeiras de Redes de Computadores, Sistemas Operacionais, Programação Orientada a Objetos e Arquitetura e Organização de Computadores.Desenvolve pesquisas na área de Machine Learning e Deep Learning, explorando as potencialidades de frameworks como Tensorflow 2.0 e Pytorch. Atualmente é professor no Instituto Federal de Educação, Ciência e Tecnologia do Tocantins e na UNITINS.